Deep Learning for Causal Inference

Bernard J. Koch[*1], Tim Sainburg[2], Pablo Geraldo Bastìas[1], Song Jiang[3], Yizhou Sun[3]
and Jacob Foster[1]

[1]UCLA Department of Sociology
[2]Harvard Medical School
[3]UCLA Department of Computer Science

April 2023

Abstract

This primer systematizes the emerging literature on causal inference using deep neural networks under the potential outcomes framework. It provides an intuitive introduction on building and optimizing custom deep learning models and shows how to adapt them to estimate/predict heterogeneous treatment effects. It also discusses ongoing work to extend causal inference to settings where confounding is non-linear, time-varying, or encoded in text, networks, and images. To maximize accessibility, we also introduce prerequisite concepts from causal inference and deep learning. The primer differs from other treatments of deep learning and causal inference in its sharp focus on observational causal estimation, its extended exposition of key algorithms, and its detailed tutorials for implementing, training, and selecting among deep estimators in Tensorflow 2 and PyTorch.

---

[*]bernardkoch@ucla.edu

Contents

Boxes

1 Introduction

This primer aims to introduce social science readers to an exciting literature exploring how deep neural networks can be used to estimate causal effects. In recent years, both causal inference frameworks and deep learning have seen rapid adoption across science, industry, and medicine. Causal inference has a long tradition in the social sciences, and social scientists are increasingly exploring the use of machine learning for causal inference (Athey and Imbens, 2016; Wager and Athey, 2018; Chernozhukov et al., 2018). Nevertheless, deep learning remains conspicuously underutilized by social scientists compared to other ML approaches, both for causal inference and more generally.

The deep learning revolution has been spurred by the flexibility and expressiveness of these models. Neural networks are nearly non-parametric and can theoretically approximate any continuous function (Cybenko, 1989), making them well suited for both classification and regression tasks. Furthermore, they can be configured with different architectures and objectives to learn from a variety of quantitative data as well as text, images, video, networks, and speech. These advantages allow them to learn vector "representations" of complex data with emergent properties. Simple examples of representation learning include the Word2Vec algorithm that discovers semantic relationship between words in texts, or face classification models that learn vectors describing facial features (Mikolov et al., 2013). More recently, generative models like DALL-E, Stable Diffusion, and ChatGPT have shown how coherent text passages and life-like images can be reconstructed from learned representations.

Here we explore the potential for leveraging these advantages to estimate causal effects. Causal inference frameworks are non-parametric, but the linear models traditionally used to estimate causal effects require strong parametric assumptions. In contrast, the nearly non-parametric nature of neural networks allows us to estimate smooth response surfaces that capture heterogeneous treatment effects for individual units with low bias.[1] The ability of these models to learn from complex data means we can extend causal inference to new settings where confounding is complicated, time-varying, or even encoded in texts, graphs, or images (see Box 1 for hypothetical examples). Lastly given the right objectives, neural networks promise to learn deconfounded representations of data, presenting a new

---

[1] Neural networks can have hundreds to billions of parameters making them effectively non-parametric. The risks of overparameterization of neural networks are discussed in Section 2.

strategy for treatment modeling.

This primer synthesizes existing literature on deep causal estimators, but it is not a review; its goals are fundamentally pedagogical and prospective rather than retrospective. In Section 2, we introduce social scientists to the fundamental concepts of deep learning, and the basic workflow for building and training their own deep neural networks within a supervised learning framework. For readers unfamiliar with causal inference, Section 3 introduces the assumptions of causal identification and three fundamental estimation strategies within the selection on observables design: matching, outcome modeling, and inverse propensity score weighting. Machine learning models often perform poorly in both theory and practice when only one of these strategies is employed, so we also introduce the concept of double robustness.

Section 4 is the main body of the article. Here we introduce four related deep learning models for the estimation of heterogeneous treatment effects: the S-learner, T-learner, TARNet and Dragonnet (Shalit et al., 2017; Shi et al., 2019). Although this literature is rapidly evolving, these four models are sufficient to illustrate how traditional estimation strategies can be used in creative ways that leverage the key strengths of neural networks (i.e., deconfounding through representation learning, semi-parametric inference). Section 5 deals with the practical considerations of building confidence intervals and interpreting neural networks. These guidelines are concretized in the companion online tutorials, which show readers how to implement and interpret the models described in Section 4 in Tensorflow 2 and PyTorch.

In Section 6, we focus on the future of deep causal information: estimators that can disentange counfounding relationships embedded within texts, images, graphs, or time-varying data. In the interest of clarity, we give hypothetical examples of the types of questions social scientists might answer with these models, and briefly describe ongoing research on each of these modalities. For fuller treatments of some of these models, see the appendix. We conclude with a discussion of how neural networks fit into the broader literature on machine learning for causal inference (Section 7).

The primer makes multiple contributions. First, it is one of the first pieces in the sociological literature to introduce the fundamentals of deep learning not only at a conceptual level (e.g, back-propagation, representation learning), but at a practical one (e.g., validation, hyperparameter tuning).

Our recommendations for training and interpreting neural networks are supported by heavily annotated tutorials that teach readers without prior familiarity with deep learning how to build their own custom models in Tensorflow 2 and PyTorch. Second, we use this foundation and select examples to build intuition on how the core strengths of deep learning can be leveraged for causal inference. Finally, we highlight future directions for this literature and argue why the future of causal estimation runs through deep learning.

Box 1: Example Scenarios for Causal Inference with Non-Traditional Data

Text. As a motivating example, Veitch et al. (2020) consider the effect of the author's reported gender (T) on the number of upvotes a Reddit post receives (Y). However gender may also "affect the text of the post, e.g., through tone, style, or topic choices, which also affects its score $[(X)]$." Controlling for a representation of the text would allow the analyst to more accurately estimate the direct effect of gender.

Images. Todorov et al. (2005) showed that split second-judgments of a politician's competence (T) from pictures (X) of their face is predictive of their electability (Y). When attempting to replicate this study using machine learning classifiers rather than human classifiers, Joo et al. (2015) suggest that the age of the face (Z) is a not-so-obvious confounder: while older individuals are more likely to appear competent, they are also more likely to be incumbents. Even if age is unknown, using neural networks to control for confounders implicitly encoded in the image (like age) could reduce bias.

Networks. Nagpal et al. (2020) explore the question of which types of prescription opioids (e.g., natural, semi-synthetic, synthetic) (T) are most likely to cause long term addiction (Y). Because of predisposition to different injuries, type of employment (X) could be a common cause of both treatment and outcome. Suppose job type is unobserved, but we know that patients are likely to associate with coworkers through homophily. To capture some of the effects of this latent unobserved confounder, analysts might choose to control for a representation of the patient's position in their social network when estimating the causal effect.

---

2 Deep Learning Fundamentals

2.1 Artificial Neural Networks

Box 2: Basic Introduction to Supervised Learning

Deep learning algorithms have most commonly been adapted for causal inference using supervised machine learning, the most popular learning framework within the field.[a] The goal of supervised learning is teach a model a non-linear function that transforms covariates/features

---

X into predicted outcomes $\hat{Y}$ in unseen data. The model learns this function from labeled examples of X_{tr} and Y_{tr} in a **training dataset**.

As in traditional statistical analyses, the function is learned by optimizing the model's parameters such that they minimize the error between its predictions $\hat{Y}_{tr}$ and the true values Y_{tr} using a **loss function** (e.g., a likelihood). In a traditional social science analysis focused on inference, we would stop here and interpret these parameters. In supervised machine learning where the focus is on generalization to unseen data, the model is ultimately used to predict outcomes Y_{te} in a **test dataset** of previously unseen covariates/features X_{te}. This framework can be generically applied to cases where Y is categorical (called classification problems), and where Y is continuous (called regression problems).

Statistical learning theory articulates the central challenge of supervised learning as a balance between **overfitting** and **underfitting** the training dataset. This is also called the **bias-variance" tradeoff**. In a regression context, bias error is the difference between the expected value of Y and the expected value of the mapping function learned by the model.[b] High bias typically results from an algorithm that has not sufficiently learned the relationships in the training dataset (i.e., underfit the data). In contrast, an algorithm that has learned the training dataset so closely that it is fitting noise in the sample (i.e., overfitting) is likely to generalize poorly, producing out-of-sample predictions with high variance. Underfitting can be easily diagnosed and addressed by increasing the complexity of the model. In the case of deep learning, model complexity can be increased by adding additional layers or parameters/neurons.

Diagnosing and addressing overfitting is a more challenging problem. In supervised learning, overfitting is diagnosed after training (but before testing) by assessing predictive performance in a reserved portion of the training set called the **validation set**. If the model fits the training dataset well but performs poorly in the validation set, it is likely to generalize poorly to the test set as well. To prevent overfitting, **regularization** techniques can be used to simplify the complexity of the model. Training and regularization of neural networks is discussed in detail in Section 2.2. For a full treatment of supervised learning and statistical learning theory, see Hastie et al. (2009).

---

[a]The other two prominent paradigms are unsupervised learning and reinforcement learning.

[b]Note that bias in statistical learning theory is not equivalent to bias of a statistical estimator.

Artificial neural networks (ANN) are statistical models inspired by the human brain (Brand et al., 2020; Goodfellow et al., 2016). In an ANN, each "neuron" in the network takes the weighted sum of its inputs (the outputs of other neurons) and transforms them using a differentiable, non-linear function (e.g. sigmoid, rectified linear unit) that outputs a value between 0 and 1 if the transformed value is above some threshold. Neurons are arrayed in layers where an input layer takes the raw data, and neurons in subsequent layers take the weighted sum of outputs in previous layers as input. An "output" layer contains a neuron for each of the predicted outcomes with transformation functions appropriate to those outcomes. For example, a regression network that predicts one outcome will

have a single output neuron without a transformation function so that it produces a real number. A regression network without any hidden layers corresponds exactly to a generalized linear model (Fig. 1A). When additional "hidden" layers are added between the input and output layers, the architecture is called a **feed-forward network** or **multi-layer perceptron** (Fig. 1B). A neural network with multiple hidden layers is called a "deep" network, hence the name "deep learning" (LeCun et al., 2015). A neural network with a single, large enough hidden layer can theoretically approximate any continuous function (Cybenko, 1989).

Box 3: Reading Machine Learning Papers: Computational Graphs and Loss Functions

    Within the machine learning literature, novel algorithms are often presented in terms of their computational graph and loss function. A computational graph (not to be confused with a causal graph) uses arrows to depicts the flow of data from the inputs of a neural network, through parameters, to the outputs. Layers of neurons or specialized sub-architectures are often generically abstracted as shapes. In our diagrams, we use purple to represent observables, orange for representation layers of the network, white for produced outputs, and red and blue for outcome modeling layers. Operations that are computed *after* prediction (i.e., for which an error gradient is not calculated) are shown with dashed lines (e.g., plug-in estimation of causal estimands).

    Along with the architecture, the loss function of a neural network is the primary means for the analyst to dictate what types of representations a neural network learns and what types of outputs it produces. In multi-task learning settings, we denote joint loss functions for an entire network as a weighted sum of the losses for substituent tasks and modules. These specific losses are weighted by hyperparameters. For example, we might weight the joint loss for a network that predicts outcomes and propensity scores as:

$$\arg\min_{h,\pi} \mathcal{L} = \mathcal{L}_h + \lambda\mathcal{L}_\pi = MSE(Y, h(X,T)) + \lambda BCE(T, \pi(X,T))$$

where $h(X,T)$ is the predicted potential outcome, $\pi(X,T)$ is the predicted propensity score, λ is a hyperparameter and MSE and BCE stand for mean squared error and binary cross entropy (i.e., log loss), common losses for regression and binary classification respectively (Box 6).

Neural networks are trained to predict their outcomes by optimizing a **loss function** (also called an objective or cost function). During training, the **backpropagation** algorithm uses calculus's chain rule to assign portions of the total error in the loss function to each neuron in the network. An optimizer, such as the stochastic gradient descent algorithm or the currently popular ADAM algorithm (Kingma and Ba, 2015), then moves each parameter in the opposite direction of this error
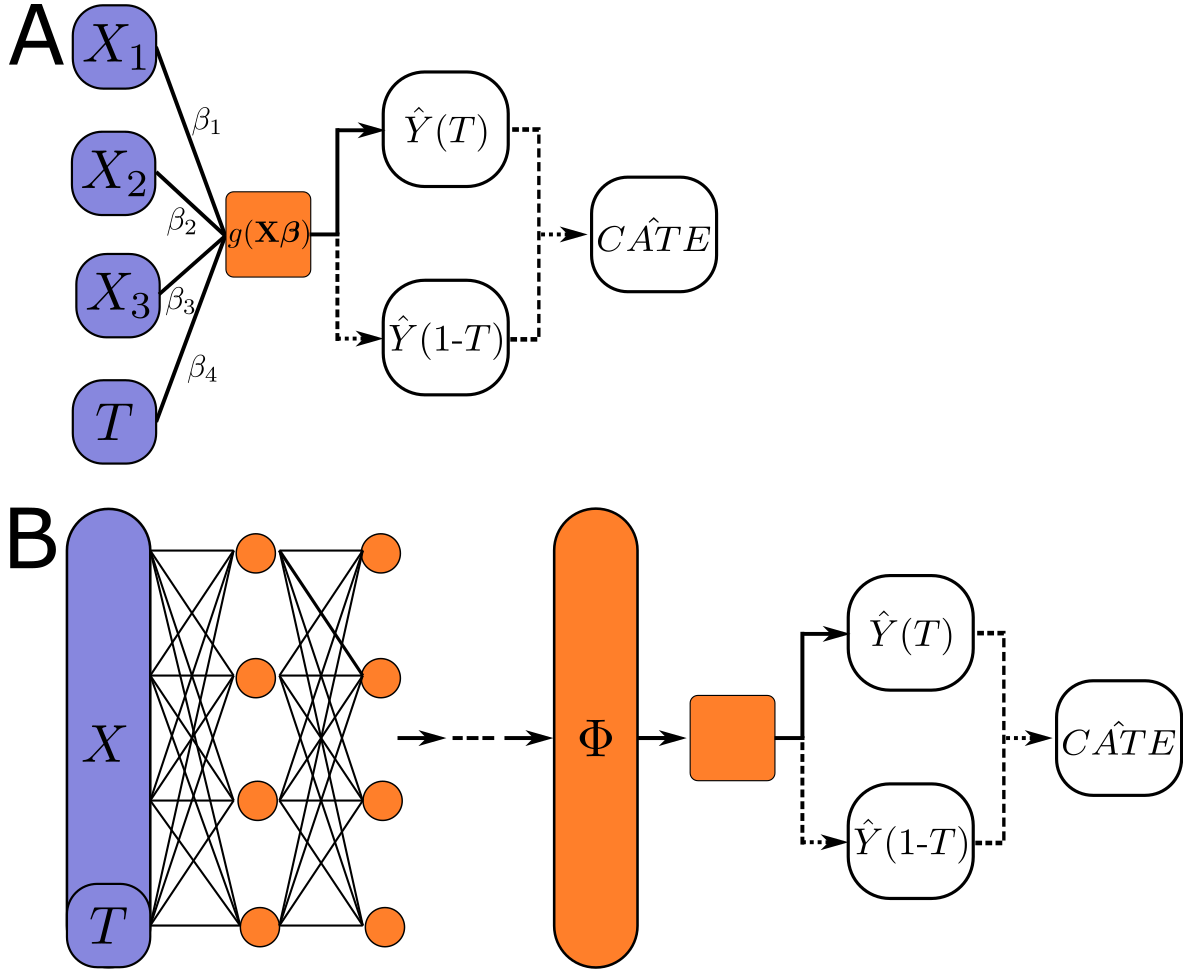
Figure 1: **A: Generalized linear model represented as a computational graph.** Observable covariates X_1, X_2, X_3 and treatment status T depicted in purple. Each of the lines between the purple inputs and the orange box represents a parameter (i.e., a β in a generalized linear model equation). The orange box is an "output neuron" that sums it's weighted inputs, performs a transformation g (the link function in GLM; in this case the identity function), and predicts the conditional outcome $\hat{Y}(T)$. Instead of theoretically interpreting these parameters from an inferential statistics perspective, machine learning approaches typically use the predicted observed and unobserved potential outcomes for plug-in estimation of causal estimands (e.g., the conditional average treatment effect $\hat{CATE}$). After training, setting T to $1 - T$ for each observation can predict the unobserved potential outcome $\hat{Y}(1 - T)$. Because this operation occurs after prediction and does not feed a gradient back to the network to optimize the parameters, it is depicted here with a dotted line. Plug-in calculation of $\hat{CATE}$ is similarly shown with a dotted line.

B: Feed-forward neural network (S-learner). In a feed-forward neural network, additional fully connected (parameterized) layers of neurons are added between the inputs and output neuron. The size of the input covariates and hidden layers are generically abstracted as boxes. The final hidden layer before the output neuron is denoted Φ because the hidden layers collectively encode a representation function (see section 2.3). In causal inference settings, this architecture is sometimes called a S(ingle)-learner because one feed-forward network learns to predict both potential outcomes.

9

gradient. Neural networks first rose to popularity in the 1980s but fell out of favor compared to other machine learning model families (e.g., support vector machines) due to their expense of training. By the late 2000s, improvements to backpropagation, advances in computing power (i.e., graphic cards), and access to larger datasets collectively enabled a deep learning revolution where ANNs began to significantly outperform other model families. Today, deep learning is the hegemonic machine learning approach in industries and fields other than social science.

2.2 Deep Learning in Practice

This section focuses on the practice of training neural networks within a supervised learning framework. While the principles behind supervised machine learning are universal, the workflow for neural networks differs substantially from other ML approaches (e.g., random forests, support vector machines) in practice. Figure 2 presents this workflow in four different parts: Set Up, Training, Model Evaluation, and Interpretation. We delve into each of these topics in more detail below. Box 2 contains a basic introduction to supervised learning for unfamiliar readers.

2.2.1 Set Up and Hyperparameters

The first step in training a neural network, as in other types of supervised machine learning, is to split your dataset into training, validation, and testing datasets (Fig. 2A). If the network is being used for statistical inference, as here, the testing dataset is optional, and inference may be conducted on just the validation set or the full dataset.

While the computational graph and loss function define a deep learning architecture (Box 3), actual implementations can vary significantly due to the choice of hyperparameters. In supervised machine learning, **hyperparameters** are parameters that are not learned automatically when training the model, but must be specified by the analyst. In deep learning, architectural hyperparameters include the number of layers to use for each section of the computational graph, the number of neurons to use in each layer, and the activation functions to be used by neurons. While some basic rules of thumb apply (e.g., use fewer layers than neurons), these choices remain poorly understood theoretically[2];

---

[2]For some interesting work on understanding neural networks theoretically from a statistical physics perspective see

1. Set Up
- Split Dataset
- Select Hyperparameters

2. Training
- Create Training Minibatches
- For all mini-batches:
  - Predict minibatch
  - Backprop error
- Calculate Validation Error
- (Epoch)

Stop when validation error rises

3. Model Evaluation
- Calculate Model Selection Criteria

Select Best Model

4. Inference and Interpretation
- Predict Test/Full Dataset
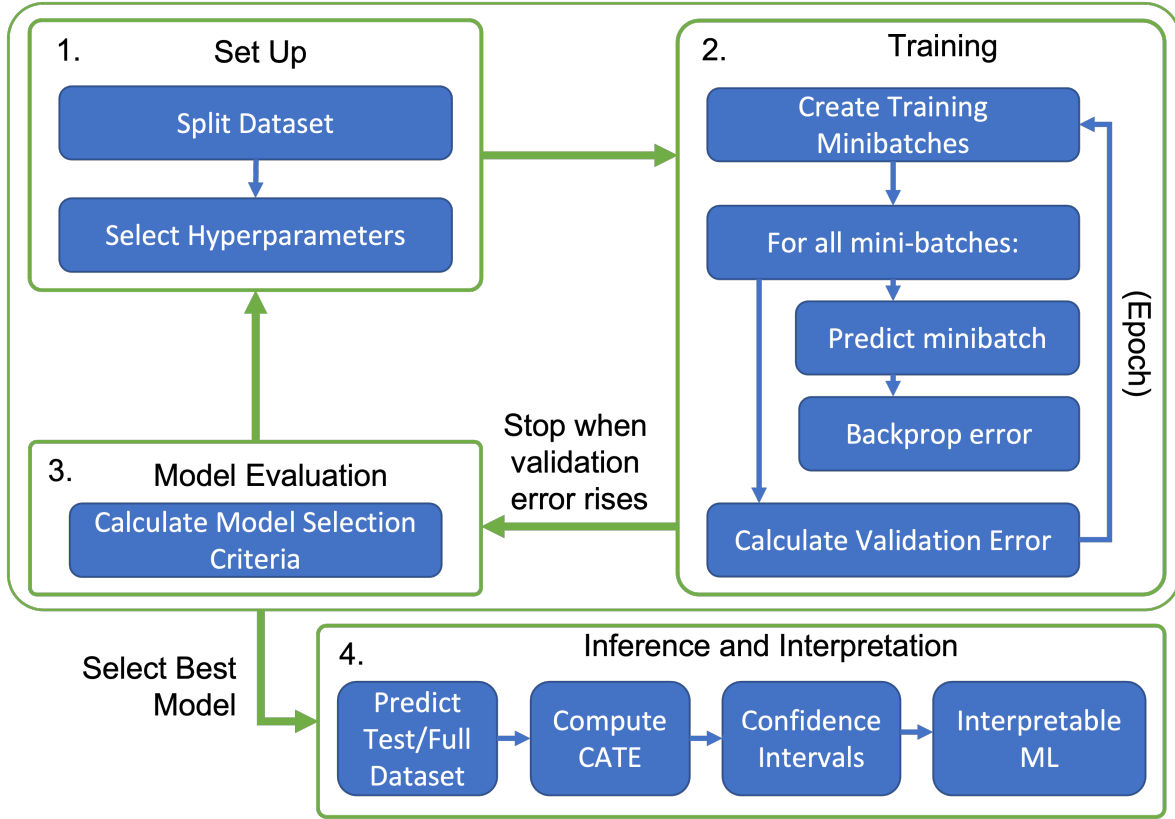- Compute CATE
- Confidence Intervals
- Interpretable ML

Figure 2: **Supervised Deep Learning Workflow. 1) Set Up:** The first step in training a deep learning model is splitting the data into a training set, validation set, and optionally a test set. Initial hyperparameters are then selected from a set of choices specified by the user. **2) Training:** In each iteration of the training process (called an epoch), the training set is randomly divided into small minibatches For each minibatch, the network makes predictions for all units, and calculates the error gradients to be assigned to each neuron in the network based on those predictions. An optimizer then move the network's parameters in the opposite direction of the error gradient. After all minibatches have been trained (one epoch), error is calculated on the entire validation set. This whole process is repeated up until the validation error stops decreasing (to avoid overfitting). **3) Model Evaluation:** A criterion (typically the validation error) is used to evaluate the performance of this hyperparameterization. New hyperparameters are then selected using a hyperparameter optimization algorithm (eg. Grid search, Bayesian hyperparameter optimization, genetic algorithms) and steps 1 and 2 are repeated. Once the hyperparameter optimization algorithm has completed its search, the "best" model is selected for inference. **4) Inference and interpretation:** With a model selected, the analyst is now ready to apply it to their test data (or in the case of statistical inference, potentially the full dataset). Predictions of the outcomes and/or propensity score can then be used to compute the CATE and calculate confidence intervals. Feature importance algorithms like SHAP or Integrated Gradients can also be used to interpret the CATE estimates.

Decisions are generally made by comparing empirical performance on the validation set, a practice called *hyperparameter tuning*.

2.2.2 Training and Regularization

Neural networks are trained by repeatedly making predictions from the training set, calculating error gradients for each parameter, and backpropagating small fractions of those error gradients. (Fig. 2 B). A full pass through examples in the training set is called a training loop or **epoch**. At the beginning of each epoch, the training set is divided into **mini-batches** of 2 to 1000 units, randomly sampled without replacement. This practice not only aids in memory management, it also improves optimization. Using small random samples reduces the risk of large "exploding" error gradients, particularly early in the training, that could cause the model to overshoot optimal solutions and instead get stuck in local minima.

The size of mini-batches can be considered a hyperparameter.[3] Because a mini-batch of data is only a sample of a sample (the training dataset), the optimizer only adjusts weight parameters by a fraction of the error gradient (the **learning rate**) to avoid overfitting. The learning rate is also a hyperparameter, that typically varies between 0.0001 and 0.01.

The non-convex nature of most loss functions[4] means that optimization often requires hundreds to potentially millions of epochs of training. Moreover, neural networks are highly susceptible to overfitting because it is easy to overparameterize them with excessive neurons/layers. To ward against overfitting, error metrics on the complete validation set are computed at the end of every epoch. In a regularization practice called "**early stopping**," analysts usually stop training once validation metrics stop improving. Other common regularization techniques include **weight decay** (i.e., ℓ^2 norm, ridge, or Tikhonov) penalties on the parameters, dropout of neurons during training, and batch normalization.

Roberts et al. (2022).

[3]In the specific context of causal inference, we recommend not having mini-batches that are too small such that the model can learn from both treated and control units with sufficient overlap.

[4]In convex functions (e.g. the OLS loss), there is a single minimum, so optimizing the function means that you will always converge at the same parameter weights. This is not the case for non-convex functions which may have many local minima.

Dropout is a regularization technique in deep learning where certain nodes are randomly silenced from training during a given epoch (Srivastava et al., 2014). The general idea of dropout is to force two neurons in the same layer to learn different aspects of the covariate/feature space and reduce overfitting. **Batch normalization** is another regularization technique applied to a layer of neurons (Ioffe and Szegedy, 2015). By standardizing (i.e. z-scoring) the inputs to a layer on a per-batch basis and then rescaling them using trainable parameters, batch normalization smooths the optimization of the loss function. The addition and extent of each of these regularization techniques can be treated as hyperparemeters.

2.2.3   Model Selection

(*Tutorial 2* [CO Open in Colab])

After the model has been trained, the analyst compares models assembled with different hyperparameterizations or initial parameter values (Fig. 2C). Hyperparameterizations can be chosen using random search, an exhaustive grid search of all possible combinations, or strategic search algorithms like Bayesian hyperparameter optimization or evolutionary optimization (Snoek et al., 2012). Validation loss metrics on the final epoch are commonly used for these comparisons.

Model selection for causal estimators is complicated by the fundamental problem of causal inference: we are not actually interested in the observed "factual" outcomes and propensity scores, but the CATE and ATE. In the case of algorithms like Dragonnet 4.3 where the validation loss explicitly targets a causal quantity, we use that as the model selection criterion. In cases where the algorithm is only trained for outcome modeling or propensity modeling, other solutions are needed. In the Appendix, we describe Johansson et al. (2020)'s proposal to use matching on a nearest neighbor approximation of the *Precision in Estimated Heterogeneous Effects* (PEHE), a measure of CATE bias, as an alternative model selection metric (Appendix A.B).

The development of more sophisticated methods for model selection of causal estimators through data simulation is an active area of research within this literature.[5] For example, Parikh et al.

---

[5]We note that crossfitting (Zivich and Breskin, 2021), another approach that has emerged for model selection of other types of machine learning causal estimators may work for the models discussed here, but is likely data-inefficient.

([2022](#)) use deep generative models to approximate the data generating distribution under weak, non-parametric assumptions. [Alaa and Van Der Schaar](#) ([2019](#)) independently model each outcome and the propensity score before using influence functions to assess model error.

2.3 Representation Learning and Multitask Learning

One comparative advantage of deep learning over other machine learning approaches has been the ability of ANNs to encode and automatically compress informative features from complex data into flexible, relevant "**representations**" or "embeddings" that make downstream supervised learning tasks easier ([Goodfellow et al.](#), [2016](#); [Bengio](#), [2013](#)). While other machine learning approaches may also encode representations, they often require extensive pre-processing to create useful features for the algorithm (i.e., feature engineering). Through the lens of representation learning, a geometric interpretation of the role of each layer in a supervised neural network is to transform its inputs (either raw data or output of previous layers) into a typically lower (but possibly higher) dimensional vector space. As a means to share statistical power, encoded representations can also be jointly learned for two tasks at once in **multi-task learning**.

The simplest example of a representation might be the final layer in a feed-forward network, where the early layers of the network can be understood as non-linearly encoding the inputs into an array of latent linear features for the output neuron ([Goodfellow et al.](#), [2016](#)) (Fig. [1](#)B). A famous example of representation learning is the use of neural networks for face detection. Examining the representations produced by each layer of these networks shows that each subsequent layer seems to capture increasingly abstract features of a face (first edges, then noses and eyes, and finally whole faces) ([LeCun et al.](#), [2015](#)). A more familiar example of representation learning to social scientists might be word vector models like Word2Vec ([Mikolov et al.](#), [2013](#)). Word2Vec is a neural network with one hidden layer and one output layer where words that are semantically similar are closer together in the representation space created by the hidden layer of the network.

The novel contribution of deep learning to causal estimation is the proposal that a neural network can learn a function Φ that produces representations of the covariates decorrelated from the treatment.
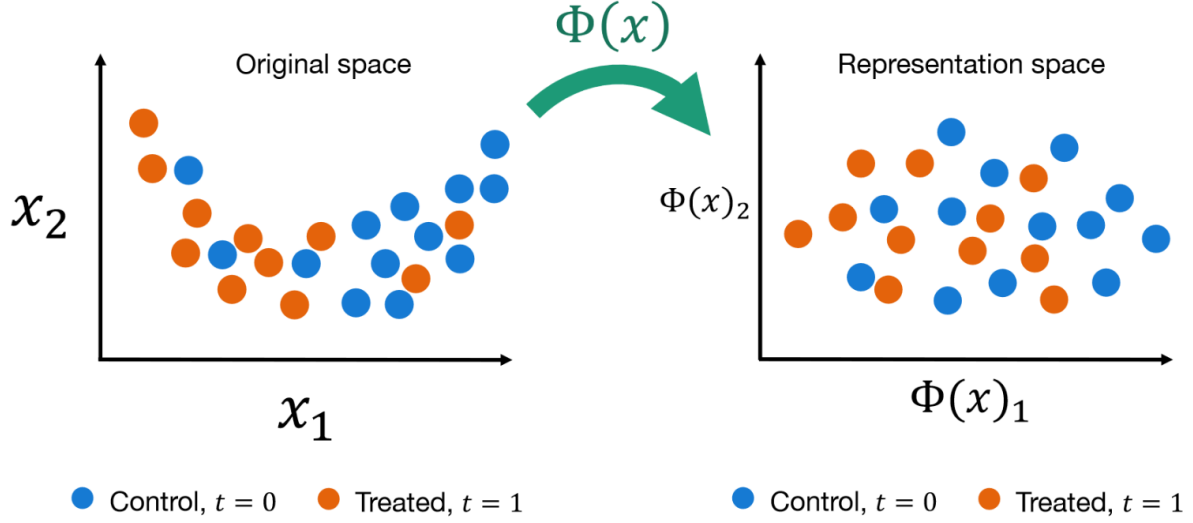
Figure 3: **Balancing through representation learning.** The promise of deep learning for causal inference is that a neural network encoding function Φ can transform the treated and control covariate distributions into a representation space such that they are indistinguishable. Used with permission from Johansson and Shen (2018).

Fundamentally, the idea is that Φ can transform the treated and control covariate distributions into a representation space such that they are indistinguishable (Fig. 3). To ensure that these representations are also still predictive of the outcome (multi-task learning), multiple loss functions are generally applied simultaneously to balance these objectives. This approach is applied in a majority of the algorithms presented in section 4.

3 Causal Identification and Estimation Strategies

3.1 Identification of Causal Effects

The papers described in this primer are primarily framed within the Potential Outcomes causal framework (Neyman-Rubin causal model) (Rubin, 1974; Imbens and Rubin, 2015). This framework is concerned with identifying the "potential outcomes" of each unit i in the sample, had it received treatment $(Y(1))$ or not received treatment $(Y(0))$. However, because each unit can only receive one treatment

regime in reality (being treated or remaining untreated), it is not possible to observe both potential outcomes for each individual (often termed "the fundamental problem of causal inference" (Holland, 1986)). While we cannot thus identify individual treatment effects $\tau_i = Y_i(1) - Y_i(0)$ for each unit, causal inference frameworks allow us to probabilistically estimate average treatment effects (ATE) and average treatment effects conditional on select covariates ($CATE$) across samples of treated and control units. Within this literature, the motivation of many papers is to present algorithms that can both infer CATEs from observational data, but also predict them for out-of-sample units where treatment status is unknown. For readers unfamiliar with causal inference, a short introduction is glossed in Box 4 with a concrete example, used in the tutorials, in Box 5.

Box 4: Basic Introduction to Causal Inference

Correlation does not equal causation, and causal statistics is concerned with the identification of causal relationships between random variables. Many causal questions we would like to ask about social data can be framed as counterfactual questions with the general format: "What would have been the outcome Y for a unit with X characteristics, if T had happened or not happened?" Equivalently, this can be reworded to "What is the causal effect of T on Y for units with characteristics X?"

Causal inference frameworks usually view randomized control trials (RCTs, also known as A/B testing in data science and industry applications)-where each unit with covariate or features X is randomly assigned to the treatment or control groups and outcome Y is subsequently measured-as the ideal approach to answering this type of question. But in many scenarios it is prohibitively expensive or unethical (e.g., randomly assigning students to attend college or not) to collect experimental data. In these cases, we can statistically adjust observational data (e.g., survey data on college attendance) to approximate the experimental ideal. The methods described in this paper are designed to answer counterfactual questions with primarily non-experimental observational data.

There are at least three different schools of causal inference that have been introduced in social statistics and econometrics (Rubin, 1974; Imbens and Rubin, 2015), epidemiology (Robins, 1986, 1987; Hernán and Robins, 2020), and computer science (Goldszmidt and Pearl, 1996; Pearl, 2009). The goal of these causal frameworks is to describe and correct for biases in data or study design that would prevent one from making a true causal claim. If these biases are correctable and the causal effect can be uniquely expressed in terms of the distribution of observed data, then we say that the causal effect is *identifiable* (Kennedy, 2016). If a causal effect is identifiable, we can use statistical tools that correct for identified biases to *estimate* the causal effect (e.g., inverse propensity score weighting, g-computation, deep learning).

The algorithms presented in this paper focus on estimating causal effects while correcting for *confounding* and *selection* bias. Loosely speaking, a confounding covariate/feature is one that is correlated with both the treatment and the outcome, misleadingly suggesting that the treatment has a causal effect on the outcome, or obscuring a true causal relationship between

---

the treatment and outcome. Often times, the confounder is a cause of the treatment *and* outcome. As an example of selection bias, estimating the causal effect of attending college (treatment) on adult income (outcome) requires controlling for the fact that parental income may be a common cause of both college attendance and adult income.

The ATE is defined as:

$$ATE = \mathbb{E}[Y_i(1) - Y_i(0)] = \mathbb{E}[\tau_i]$$

where $Y(1)$ and $Y(0)$ are the potential outcomes had the unit i received or not received the treatment, respectively. The CATE is defined as,

$$CATE = \mathbb{E}[Y_i(1) - Y_i(0)|X_i = x] = \mathbb{E}[\tau_i|X_i = x]$$

where X is the set of selected, observable covariates, and $x \in X$.

Box 5: Applied Causal Inference Example: The Infant Health and Development Study

To make this problem setting more concrete for readers unfamiliar with causal inference, consider simulations based on the 1985-1988 Infant Health and Development Study that are widely used as benchmarks within this literature. In this experiment, premature children were randomly assigned to intensive, high-quality childcare (T), and their cognitive test scores were measured later (Y). The authors also measured numerous other covariates X including "pregnancy complications, child's birth weight and gestation age, birth order, child's gender, household composition, day care arrangements, source of health care, quality of the home environment, parents' race and ethnicity, and maternal age, education, IQ, and employment" (Ramey et al., 1992). The ATE would be the effect of intensive child care on cognitive scores across all children, while various $CATE$s might be formulated to better understand how the effects of child care vary for female children, children born to teenage mothers, or children with unemployed parents.

Hill (2011) turns this experimental data into an observational benchmark by re-simulating the outcome such that the covariates X induce confounding bias between the treatment and outcome. While the simulations don't preserve the names of the covariates, we can imagine some confounding relationships that might be present in an observational study. For example, suppose that affluent (X_1) parents are more likely able to afford high-quality child care (T), but there is actually a weak association between childcare and premature babies' cognitive ability (Y). We also know affluent parents are more likely to engage in breastfeeding (X_2), which is positively associated with higher cognitive ability (Heck et al., 2006; Kramer et al., 2008). If

---

we do not account for the correlation between income and childcare ($X_1 \rightarrow T$), or income and cognitive ability ($X_1 \rightarrow X_2 \rightarrow Y$), we may have bias in our $ATE/CATE$ estimates, or worse, erroneously interpret the correlation between childcare and cognitive ability as causal. This example is depicted in a causal graph below.
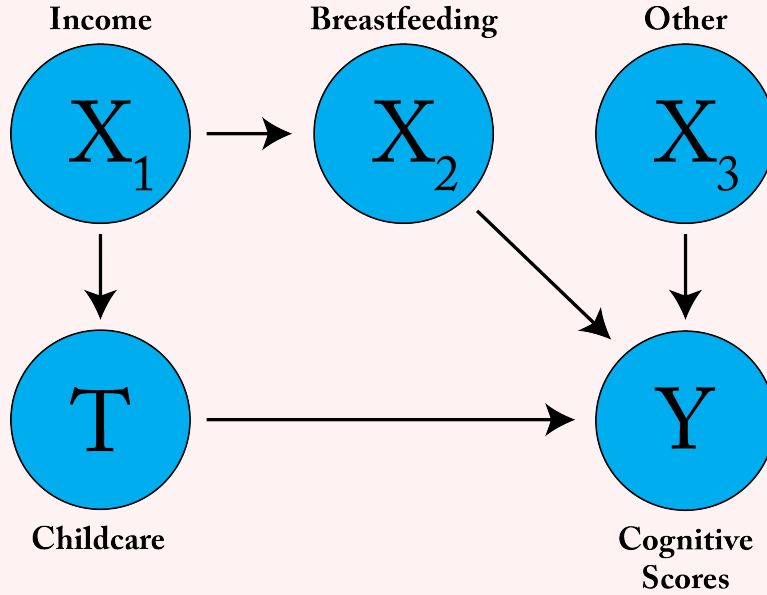
The hypothetical confounding bias presented here can be adjusted for either through treatment modeling (e.g., inverse propensity score weighting, non-parametric, deep representation learning) to block the path $X_1 \rightarrow T$, outcome modeling (e.g., generalized linear models, deep regression) to block the path $X_1 \rightarrow X_2 \rightarrow Y$, or both (see Section 3.2). For coded examples using many of these approaches in Tensorflow and Pytorch on the IHDP benchmark, please see the tutorials.

Within the machine learning literature on causal inference treated here, the primary strategy for causal identification is **selection on observables**. A challenge to identifying causal effects is the presence of confounding relationships between covariates associated with both the treatment and the outcome.

The key assumption allowing the identification of causal effects in the presence of confounders is:

1. **Conditional Ignorability/Exchangability** The potential outcomes $Y(0)$, $Y(1)$ and the treat-

ment T are conditionally independent given X,

$$Y(0), Y(1) \perp\!\!\!\perp T | X$$

Conditional Ignorability specifies that there are no unmeasured confounders that affect both treatment and outcome outside of those in the observed covariates/features X. Additionally X may contain predictors of the outcome, but should not contain instrumental variables or colliders within the conditioning set.[6]

Other standard assumptions invoked to justify causal identification are:

2. **Consistency/Stable Unit Treatment Value Assumption (SUTVA)**. Consistency specifies that when a unit receives treatment, their observed outcome is exactly the corresponding potential outcome (and the same goes for the outcomes under the control condition). Moreover, the response of any unit does not vary with the treatment assignment to other units (i.e., no network or spillover effects), and the form/level of treatment is homogeneous and consistent across units (no multiple versions of the treatment). More formally,

$$T = t \rightarrow Y = Y(T)$$

3. **Overlap**. For all $x \in X$ (i.e., any observed covariate value), all treatments $t \in \{0, 1\}$ have a non-zero probability of being observed in the data, within the "strata" defined by such covariates,

$$1 > p(T = t | X = x) > 0$$

4. An additional assumption often invoked at the interface of identification and estimation using neural networks is:

Invertability
$$\Phi^{-1}(\Phi(X)) = X$$

---

[6]A variable is a collider if it is caused by two other variables. Controlling for colliding variables, or descendants of colliding variables will induce a spurious correlation between the parents.

In words, there must exist an inverse function of the representation function Φ encoded by a neural network that can reproduce X from representation space. This is required for the Conditional Ignorability assumption to hold when using representation learning. From a practical perspective, it also means that the representation we created is rich enough to capture the causal relationships we are interested in.

For reference, we describe the full notation used within the review in Box 6.

Box 6: Notation for Causal Inference and Estimation

We use uppercase to denote general quantities (e.g., random variables) and lowercase to denote specific quantities for individual units (e.g., observed variable values).

Causal identification

- Observed covariates/features: X
- Potential outcomes: $Y(0)$ and $Y(1)$
- Treatment: T
- Unobservable Individual Treatment Effect: $\tau_i = Y_i(1) - Y_i(0)$
- Average Treatment Effect: $ATE = \mathbb{E}[Y_i(1) - Y_i(0)] = \mathbb{E}[\tau_i]$
- Conditional Average Treatment Effect: $CATE(x) = \mathbb{E}[Y_i(1) - Y_i(0)|X_i = x] = \mathbb{E}[\tau_i|X_i = x]$

Deep learning estimation

- Predicted potential outcomes: $\hat{Y}(0)$ and $\hat{Y}(1)$
- Outcome modeling functions: $\hat{Y}(T) = h(X, T)$
- Propensity score function: $\pi(X, T) = P(T|X)$ (where $\pi(X, 0) = 1 - \pi(X, 1)$)
- Representation functions: $\Phi(X)$ (producing representations ϕ)
- Loss functions: $\mathcal{L}(true, predicted)$
- Loss abbreviations: MSE (mean squared error), BCE (binary cross-entropy), CCE (categorical cross-entropy)
- Loss hyperparameters: λ, α, β
- Estimated CATE*: $C\hat{A}TE_i = \hat{\tau}_i = \hat{Y}_i(1) - \hat{Y}_i(0) = h(X, 1) - h(X, 0)$
- Estimated ATE: $\hat{ATE} = \frac{1}{N} \sum_{i=1}^{N} \hat{\tau}_i$

Beyond the ATE and $CATE$ there is an additional metric commonly used in the machine learning literature, first introduced by Hill (2011) called the *Precision in Estimated Heterogeneous Effects (PEHE)*. PEHE is the average error across the predicted $CATE$s.

- Precision in Estimated Heterogeneous Effects: $PEHE = \frac{1}{N} \sum_{i=1}^{N} (\tau_i - \hat{\tau}_i)^2$

Beyond being a metric for simulations with known counterfactuals, the $PEHE$ has theoretical significance in the formulation of generalization bounds within this literature (Shalit et al., 2017; Johansson et al., 2018, 2020; Zhang et al., 2020).

*Note that we use $\hat{\tau}$ to refer to the estimated CATE because truly individual treatment effects cannot be described only by the observed covariates X.

→ What about confounding in RCTs ?

3.2 Estimation of Causal Effects

Once a strategy for identifying causal effects from available data has been developed (arguably the harder and more important part of causal inference), statistical methods can be used to estimate causal effects by controlling for confounding bias. There are two fundamental approaches to estimation: **treatment modeling** to control for correlations between the covariates X and the treatment T, and **outcome modeling** to control for correlations between the treatment X and the outcome Y (Fig. 4). Below we briefly review three traditional techniques for removing confounding bias to motivate our systematization of deep learning models. First, we discuss outcome modeling through regression. Next, we consider treatment modeling through non-parametric matching. Finally, we discuss treatment modeling through inverse propensity score weighting (IPW) and introduce the concept of double robustness.

3.2.1 Outcome Modeling: Regression

Assuming the treatment effect is constant across covariates/features or the probability of treatment is constant across all covariates/features (both improbable assumptions), the simplest consistent approach to estimating the ATE is to regress the outcome on the treatment indicator and covariates using a linear model.[7] The ATE is then the coefficient of the treatment indicator. Without loss of generality, we call outcome models of this nature, linear or non-linear, h:

$$\hat{Y}_i(T) = h(X_i, T)$$

A slightly more sophisticated semi-parametric approach to **outcome modeling**, used widely in the application of machine learning to causal inference, is to use $h(X, T)$ to impute $\hat{Y}(1)$ and $\hat{Y}(0)$, and calculate the CATE for each unit as a plug-in estimator:

$$\widehat{CATE}_i = \hat{\tau}_i = \hat{Y_i(1)} - \hat{Y_i(0)} = h(X_i, 1) - h(X_i, 0)$$

[7]Another outcome modeling approach that could be used to estimate the outcome, not discussed here, is g-computation (Robins, 1986; Hernán and Robins, 2020).

⇒ Look more into treatment vs outcome modelling.

Treatment Modeling
  -IPW
  -Matching
  -Representation
   Learning

X

Outcome Modeling
  -GLM
  - Maching Learning

T

Y

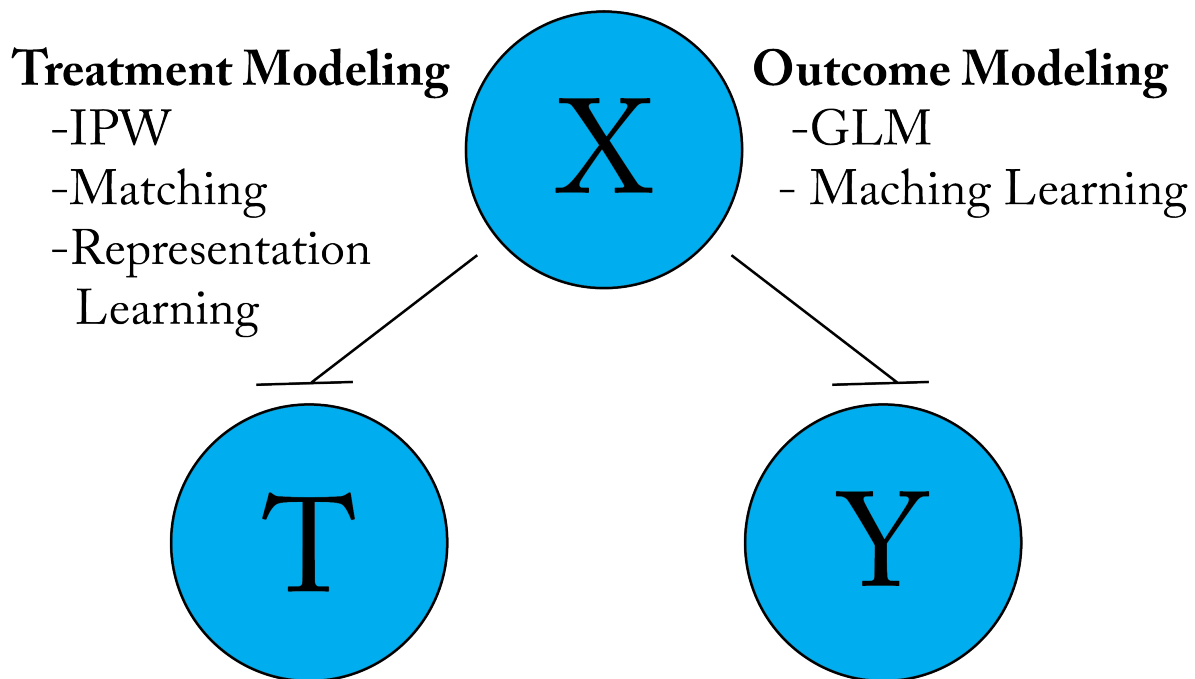Figure 4: **Two fundamental approaches to deconfounding.** Blunted arrows indicate blocked causal paths. Treatment modeling approaches like inverse propensity weighting, balancing, and representation learning adjust for the association between the covariates X and the treatment T. Outcome modeling approaches like generalized linear models or machine learning regressors adjust for the association between X and the outcome Y.

and the ATE as:

$$\widehat{ATE} = \frac{1}{N} \sum_{i=1}^{N} \hat{\tau}_i$$

3.2.2 Treatment Modeling: Non-Parametric Matching

A common treatment-modeling strategy is balancing the treated and control covariate distributions through matching. Matching requires the analyst to select a distance measure that captures the difference in observed covariate distributons between a treated and untreated unit (Austin, 2011). Units with treatment status T can then be matched with one or more counterparts with treatment status $1-T$ using a variety of algorithms (Stuart, 2010). In a one-to-one matching scenario where each treated unit has an otherwise identical untreated counterpart, the covariate distribution of treated and control units is indistinguishable.

3.2.3 Treatment Modeling: Inverse Propensity Score Weighting

Another common approach is **inverse propensity score weighting (IPW)**. In IPW, units are weighted on their inverse propensity to receive treatment. Without loss of generality, we call the propensity function π. The propensity score is calculated as the probability of receiving treatment conditional on covariates:

$$\pi(X, T) = P(T|X)$$

The simplest IPW estimator of the ATE is then:

$$\widehat{ATE} = \frac{1}{N} \sum_{i=1}^{N} \left\{ \frac{T_i Y_i}{\hat{\pi}(X_i, 1)} + \frac{(1 - T_i) Y_i}{\hat{\pi}(X_i, 0)} \right\} \tag{1}$$

Note that only one of the two terms is active for any given unit. Furthermore, this presentation looks different than how the IPW is generally presented because we use π as a function with different

outputs depending on the value of T rather than a scalar (Box 6).[8]

IPW weighting is attractive because if the propensity score π is specified correctly, it is an unbiased estimator of the ATE. Moreover, the IPW is consistent if π is estimated consistently (Rosenbaum and Rubin, 1983; Glynn and Quinn, 2010).

3.2.4 Double Robustness

Because different models make different assumptions, it is not uncommon to combine outcome modeling with propensity modeling or matching estimators to create **doubly-robust** estimators. For example, one of the most widely used doubly-robust estimators is the Augmented-IPW (AIPW) estimator.

$$A\hat{T}E = \frac{1}{N} \sum_{i=1}^{N} [\underbrace{(\frac{T}{\pi(\Phi(X),1)} - \frac{1-T}{\pi(\Phi(X),0)})}_{\text{Treatment Modeling}} \times \underbrace{[Y - h(\Phi(X),T)]}_{\text{Residual Confounding}} + \underbrace{[h(\Phi(X),1) - h(\Phi(X),0)]}_{\text{Outcome Modeling}} \quad (2)$$
$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{Adjustment}}$$

The first term is the difference in prediction from two outcome models, one for treated and one for control units, while the last terms is a "corrected" IPW estimator replacing the raw outcome by the residuals from the regression models. As expected, this estimator is unbiased if the IPW and regression estimators are consistently estimated. However, the model is attractive because it will be consistent if *either* the propensity score $\pi(X,T)$ is correctly specified or the regression model h is consistently specified (Glynn and Quinn, 2010). The model also provide efficiency gains with respect to the use of each model separately, and especially with respect to weighting alone.

Doubly robust estimation is especially important for causal estimation using machine learning. When using simple outcome plug-in estimators, bias is directly dependent on estimation error, which may be different for each potential outcome depending on the modeling strategy (Kennedy, 2020). Machine learning estimation of the propensity score can also rely heavily on non-confounding predictors, giving rise to extreme weights (Schnitzer et al., 2016). More generally, there are no asymptotic

---

[8]To de-emphasize the contribution of units with extreme weights due to sparse data, sometimes a "stabilized" IPW is used (Glynn and Quinn, 2010).

linearity guarantees for machine learning estimators which may converge at a slow rate, leading to misleading confidence intervals (Naimi and Balzer, 2018; Zivich and Breskin, 2021). For these reasons, plug-in machine learning estimation often has poor empirical performance when not using double robust estimators (Benkeser et al., 2017; Bodnar et al., 2022; Kennedy, 2020; Zivich and Breskin, 2021).

The growth of machine learning for causal inference literature has thus been largely driven by the introduction of semi-parametric frameworks. Semi-parametric frameworks address these issues by using machine learning only to estimate the nuissance parameters (i.e., potential outcomes and propensity score) of influence functions for causal parameters like the ATE and CATE (Chernozhukov et al., 2018; Kennedy, 2016; Van der Laan and Rose, 2011). In these approaches, the estimation of causal parameters is only-second order dependent on machine learning error, there is double-robustness against inconsistent estimation, and guarantees of fast convergence and asymptotically-valid confidence intervals even if the machine learning models converge slowly (Benkeser et al., 2017; Kennedy, 2020; Naimi and Balzer, 2018; Zivich and Breskin, 2021). We use the final algorithm introduced below, Dragonnet, as an opportunity to provide an intuitive introduction to semi-parametric theory and how it can be used for doubly robust estimation (Shi et al., 2019).

4   Three Different Approaches to Deep Causal Estimation

The architectures proposed in the deep learning literature for causal estimation build upon the core idea discussed above. First, we introduce "S-Learners" and "T-Learners" to show how neural networks can be used to estimate non-linearities in potential outcomes. Second, given the right objectives, a neural network can learn representations of the treated and control distributions that are deconfounded (Fig. 3). This approach, which can be related theoretically to non-parametric matching, is illustrated by the foundational TARNet algorithm in section 4.3 (Shalit et al., 2017). Finally, the machine learning for causal inference literature has been largely driven by the introduction of semi-parametric frameworks that allow predictive machine learning models to be plugged-in to doubly robust estimation equations (Van der Laan and Rose, 2011; Chernozhukov et al., 2018, 2021). In section 4.3, we introduce the

concept of influence functions and the targeted maximum likelihood estimator to explain the Dragonnet algorithm. For clarity the algorithms presented here all share a familial resemblence to the TARNet algorithm. However, we note that there are many other approaches to using deep learning for causal inference (e.g., the generative models described in Appendix A.D).

4.1 Deep Outcome Modeling

S-Learners and T-Learners (Tutorial 1 CO Open in Colab *)*

Because at most one potential outcome is unobserved, it is not possible to apply supervised models to directly learn treatment effects. Across econometrics, biostatistics, and machine learning, a common approach to this challenge has been to instead use machine learning to model each potential outcome separately and use plug-in estimators for treatment effects (Chernozhukov et al., 2018; Van der Laan and Rose, 2011; Wager and Athey, 2018). As with linear models, a single neural model can be trained to learn both potential outcomes ("S[ingle]-learner") (Fig. 1B), or two independent models can be trained to learn each potential outcome (a "T-learner") (Johansson et al., 2020) (Fig. 5A). In both cases, the neural network estimators would be feed-forward networks tasked with minimizing the MSE in the prediction of observed outcomes. In a slight abuse of notation, the joint loss function for a T-learner can be written as:

$$\mathcal{L}(Y, h(X, T)) = MSE(T_i(Y_i, h_1(X_i, 1)) + (1 - T_i)(Y_i, h_0(X_i, 0)) \tag{3}$$

where h_1 and h_0 represent separate networks for each potential outcome.

After training, inputting the same unit into both networks of a T-learner will produce predictions for both potential outcomes: $\hat{Y}(T)$ and $\hat{Y}(1 - T)$. We can plug-in these predictions to estimate the $CATE$ for each unit,

$$\hat{\tau}_i = (1 - 2T_i)(\hat{Y}_i(1 - T_i) - \hat{Y}_i(T_i))$$

where the first term is a switch to make sure the treated potential outcome comes first. The average
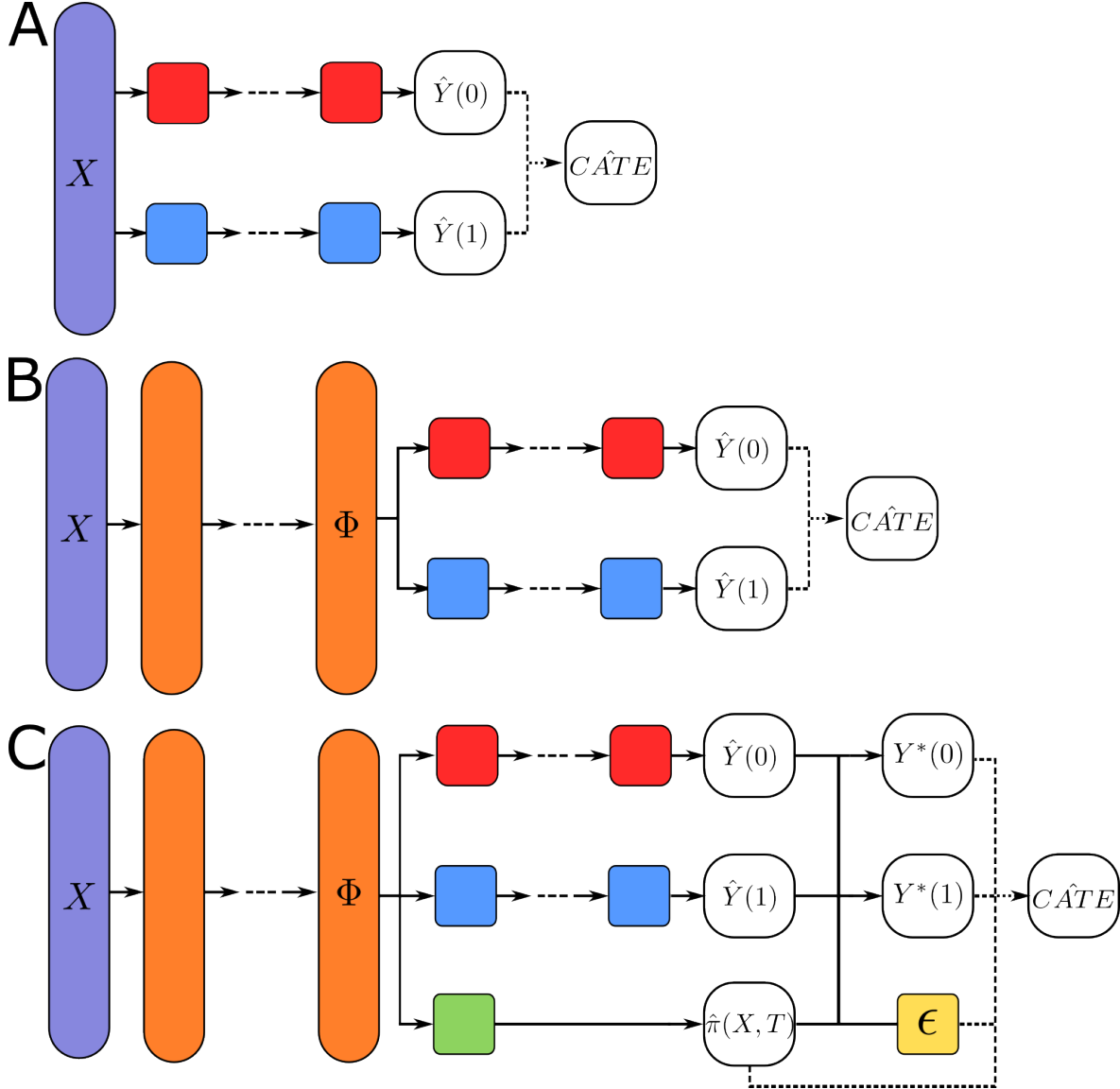
Figure 5: A. **T-learner.** In a T-learner, separate feed-forward networks are used to model each outcome. We denote the function encoded by these outcome modelers h. B. **TARNet.** TARNet extends the T-learner with shared representation layers. The motivation behind TARNet (and further elaborations of this model) is that the multi-task objective of accurately predicting both the treated and control potential outcomes forces the representation layers to learn a balancing function Φ such that the $\Phi(X|T=0)$ and $\Phi(X|T=1)$ are overlapping distributions in representation space. For a code implementation, see Box 7. **C. Dragonnet** Dragonnet also adds a propensity score head to TARNet and a free "nudge" parameter ϵ. In an adaptation of Targeted Maximum Likelihood Estimation, $\hat{\pi}$ and ϵ are used to re-weight the outcomes to provide lower biased estimates of the ATE.

treatment effect as,

$$\widehat{ATE} = \frac{1}{N} \sum_{i=1}^{N} \hat{\tau}_i$$

Nearly all of the models described below combine this plug-in outcome modeling approach with other forms of treatment adjustment.

4.2 Balancing through Representation Learning

TARNet (Tutorial 1 [CO Open in Colab] *)*

Balancing is a treatment adjustment strategy that aims to deconfound the treatment from outcome by forcing the treated and control covariate distributions closer together (Johansson et al., 2016). *The novel contribution of deep learning to the selection on observables literature is the proposal that a neural network can transform the covariates into a representation space Φ such that the treated and control covariate distributions are indistinguishable (Fig. 3).*

To encourage a neural network to learn balanced representations, the seminal paper in this literature, Shalit et al. (2017), proposes a simple two-headed neural network called Treatment Agnostic Regression Network (TARNet) that extends the outcome modeling T-learner with shared representation layers (Fig. 5B). Each head models a separate potential outcome: one head learns the function $\hat{Y}(1) = h_1(\Phi(X), 1)$, and the other head learns the function $\hat{Y}(0) = h_0(\Phi(X), 0)$. During training, only one head will receive error gradients at a time (the one predicting the observed outcome). However, both heads backpropagate their gradients to shared representation layers that learn $\Phi(X)$. The idea is that these representation layers must learn to balance the data because they are tasked with predicting both outcomes. The authors of this algorithm have subsequently extended TARNet with additional losses in an algorithm called CFRNET that explicitly encourage balancing by minimizing a statistical distance between the two covariate distributions in representation space (see Appendix A.A for details) (Johansson et al., 2018, 2020).

The complete objective for the network is to fit the parameters of h and Φ for all n units in the

28

training sample such that,

$$\underset{h,\Phi}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} (Y_i - (T_i(Y_i, \underbrace{h_1(\Phi(X_i), 1)}_{\hat{Y}_i(1)}) + (1 - T_i)(Y_i, \underbrace{h_0(\Phi(X_i), 0)}_{\hat{Y}_i(0)}))^2 + \lambda \underbrace{\mathcal{R}(h)}_{L_2} \qquad (4)$$

or more compactly,

$$\underset{h,\Phi}{\arg\min} MSE(Y_i, \underbrace{h(\Phi(X_i), T_i)}_{\hat{Y}_i(T_i)}) + \lambda \underbrace{\mathcal{R}(h)}_{L_2} \qquad (5)$$

where $\mathcal{R}(h)$ is a model complexity term (e.g., for L_2 regularization) and λ is a hyperparameter chosen through model selection. For coded versions of TARNet in Tensorflow and Pytorch, see Box 7.

Box 7: TARNet in Code

   Below we show simple implementations of TARNet in Python Tensorflow 2 and Pytorch. For more explanation on this implementation and to run this code on the IHDP data, see the tutorials.
   Tensorflow 2 Functional API (Keras)

```
def make_tarnet(input_dim):
    #The argument is the number of X covariates.
    x = Input(shape=(input_dim,), name='input')

    #In TF fxnl API, stack layers by feeding output of prev layer to next
    #Make 2 representation layers
    #units is the output dim of layer
    #elu is \"exponentiated linear unit" activation fxn
    phi = Dense(units=200, activation='elu')(x)
    phi = Dense(units=200, activation='elu')(phi)

    #Begin separate outcome modeling heads
    y0_hidden = Dense(units=100, activation='elu')(phi)
    y1_hidden = Dense(units=100, activation='elu')(phi)

    # Add second layers
    y0_hidden = Dense(units=100, activation='elu')(y0_hidden)
    y1_hidden = Dense(units=100, activation='elu')(y1_hidden)

    # Output predictions
    y0_pred = Dense(units=1, activation=None)(y0_hidden)
    y1_pred = Dense(units=1, activation=None)(y1_hidden)

    #Bundle outputs
    concat_pred = Concatenate(1)([y0_pred, y1_pred])

    #instantiate model
    model = Model(inputs=x, outputs=concat_pred)
    return model
```

```python
class TARNet(nn.Module):
    def __init__(self,input_dim):
        super(TARNet,self).__init__()

        self.phi = nn.Sequential(
            #both input and output dims are specified in torch
            nn.Linear(input_dim, 200),
            nn.ELU(), #activations are discrete from layers
            nn.Linear(200,200),
            nn.ELU())

        self.y0_hidden = nn.Sequential(
            nn.Linear(200, 100),
            nn.ELU(),
            nn.Linear(100,100),
            nn.ELU())

        self.y1_hidden = nn.Sequential(
            nn.Linear(200, 100),
            nn.ELU(),
            nn.Linear(100,100),
            nn.ELU())

        self.y0_pred =nn.Linear(100,1)
        self.y1_pred = nn.Linear(100,1)

    #the flow of data/gradients in torch is declared in a forward fxn
    def forward(self,X):
        rep = self.phi(X)
        y0_rep=self.y0_hidden(rep)
        y0_hat=rep=self.y0_pred(y0_rep)

        y1_rep=rep=self.y1_hidden(rep)
        y1_hat=rep=self.y1_pred(y1_rep)

        return y0_hat, y1_hat
```

4.3 Double Robustness with Inverse Propensity Score Weighting

Rather than applying losses directly to the representation function, IPW methods estimate propensity scores from representations using the function $\pi(\Phi(X), T) = P(T|\Phi(X))$. As in traditional IPW estimators, these methods exploit the sufficiency of correctly-specified propensity scores to reweight the plugged-in outcome predictions and provide unbiased estimates of the ATE (Rosenbaum and Rubin, 1983). Because these models combine outcome modeling with IPW, they retain the attractive statistical properties of doubly robust estimators discussed in section 3.2.2 (Atan et al., 2018). In this

section we focus on Shi et al. (2019)'s Dragonnet model, which adapts semi-parametric estimation theory for batch-wise neural network training in a procedure they call "Targeted Regularization" (TarReg) (Kennedy, 2016). Given the increasing importance of semi-parametric theory and "double machine learning" across the causal estimation literature, we include a brief introduction to semi-parametric theory and targeted maximum likelihood estimation (TMLE) before diving into the details of the Dragonnet algorithm Van der Laan and Rose (2011); Chernozhukov et al. (2018).

Dragonnet (Tutorial 3 CO Open in Colab *)*

A trivial extension to TARNet is to add a third head to predict the propensity score. This third head could use multiple neural network layers or just a single neuron, as proposed in Dragonnet (Fig. 5C) (Shi et al., 2019). Dragonnet uses this additional head to develop a training procedure called "Targeted Regularization" for semi-parametric causal estimation, inspired by "Targeted Maxmimum Likelihood Estimation" (TMLE)(Van der Laan and Rose, 2011).

With three heads, the basic loss function for this network looks like:

$$\underset{\Phi,\pi,h}{\arg\min} \underbrace{MSE(Y_i, h(\Phi(X_i), T_i))}_{\text{Outcome Loss}} + \alpha \underbrace{\text{BCE}(T_i, \pi(\Phi(X_i), T_i))}_{\pi \text{ Loss}} + \lambda \underbrace{\mathcal{R}(h)}_{L_2} \tag{6}$$

with α being a hyperparameter to balance the two objectives. The mean squared error and binary cross-entropy are standard objective functions in machine learning for regression and binary classification, respectively. Note that the first term is simply an expansion of the first term in equation 4.2

Below, we explore how the authors add a second loss on top of this one to allow for semi-parametric estimation.

4.3.1   Semi-parametric Theory of Causal Inference

In recent years, semi-parametric theory has emerged as a dominant theoretical framework for applying machine learning algorithms, including neural networks, to causal estimation (Chernozhukov et al., 2018, 2021, 2022; Farrell et al., 2021; Kennedy, 2016; Nie and Wager, 2021; Van der Laan and Rose, 2011; Wager and Athey, 2018). The great appeal of these frameworks is that they allow for ma-

chine learning algorithms to be plugged-in for non-linear estimates of outcomes and propensity score, while still providing attractive statistical guarantees (e.g., consistency, efficiency, asymptotically-valid confidence intervals).

At a very intuitive level, semi-parametric causal estimation is focused on estimating a target parameter of a distribution P (the ATE) of treatment effects $T(P)$ (Fisher and Kennedy, 2021). While we do not know the true distribution of treatment effects because we lack counterfactuals, we do know some parameters of this distribution (e.g., the treatment assignment mechanism). We can encode these constraints in the form of a likelihood that parametrically defines a set of possible approximate distributions $\mathcal{P}$ from our existing data P. Within this set there is a sample-inferred distribution $\tilde{P} \in \mathcal{P}$, that can be used to estimate $T(P)$ using $T(\tilde{P})$.

Regardless of $\tilde{P}$ chosen, $\tilde{P} \neq P \rightarrow T(\tilde{P}) \neq T(P)$. We do not know how to pick $\tilde{P}$ with finite data to get the best estimate $T(\tilde{P})$. We can maximize a likelihood function to pick $\tilde{P}$, but there may be "nuisance" parameters in the likelihood that are not the target and we do not care about estimating accurately. Maximum likelihood optimization may provide lower-biased estimates of these nuissance terms at the cost of better estimates of $T(P)$.

To sharpen the likelihood's focus on $T(P)$, we define a "nudge" parameter ϵ that moves $\tilde{P}$ closer to P (thus moving $T(\tilde{P})$ closer to $T(P)$). An influence curve of $T(P)$ tells us how changes in ϵ will induce changes in $T(P + \epsilon(\tilde{P} - P))$. We'll use this influence curve to fit ϵ to get a better approximation of $T(P)$ within the likelihood framework. In particular, there is a specific **efficient influence curve (EIC)** that provides us with the lowest variance estimates of $T(P)$. In causal estimation, solving the EIC for the ATE yields estimates that are asymptotically unbiased, efficient, and have confidence intervals with (asymptotically) correct coverage.

The EIC for the ATE is,

$$EIC_{ATE} = \frac{1}{N} \sum_{i=1}^{N} [(\underbrace{\frac{T_i}{\pi(X_i, 1)} - \frac{1 - T_i}{\pi(X_i, 0)}}_{\text{Treatment Modeling}}) \times \underbrace{(Y_i - h(X_i, T))}_{\text{Residual Confounding}}] + \underbrace{[h(X_i, 1) - h(X_i, 0)]}_{\text{Outcome Modeling}}] - ATE \quad (7)$$

$$\underbrace{}_{\text{Adjustment}}$$

Setting EIC_{ATE} to it's mean of 0,

$$ATE = \frac{1}{N} \sum_{i=1}^{N} [(\underbrace{\frac{T_i}{\pi(X_i, 1)} - \frac{1 - T_i}{\pi(X_i, 0)}}_{\text{Treatment Modeling}}) \times \underbrace{(Y_i - h(X_i, T))}_{\text{Residual Confounding}}] + \underbrace{[h(X_i, 1) - h(X_i, 0)]}_{\text{Outcome Modeling}}]] \tag{8}$$

<center>Adjustment</center>

The underbraces illustrate how EIC_{ATE} resembles a doubly robust estimator. When the EIC is minimized (set to 0) as in equation 8, the ATE is equal to the outcome modeling estimate plus a treatment modeling estimate proportional to the residual error.

4.3.2   From TMLE to Targeted Regularization

Targeted Regularization (TarReg) is closely modeled after "Targeted Maxmimum Likelihood Estimation" (TMLE) (Van der Laan and Rose, 2011). TMLE is an iterative procedure where a nuissance parameter ϵ is used to nudge the outcome models towards sharper estimates of the ATE when minimizing the EIC as in Equation 8.[9]

1. Fit h by predicting outcomes (e.g., using TARNet) and minimizing $MSE(Y, h(\Phi(X), T))$

2. Fit π by predicting treatment (e.g., using logistic regression) and $BCE(T, \pi(\Phi(X), T))$

3. Plug-in h and π functions to fit ϵ and estimate $h^*(X, T)$ where,

$$\underbrace{h^*(X_i, T_i)}_{Y^*} = \underbrace{h(\Phi(X_i), \Phi(T_i))}_{\hat{Y}} + \underbrace{\left(\frac{T_i}{\pi(\Phi(X_i), 1)} - \frac{1 - T_i}{\pi(\Phi(X_i), 0)} \right)}_{\text{Treatment Modeling Adjustment}} \times \underbrace{\epsilon}_{\text{"nudge"}}$$

by minimizing $MSE(Y, h^*(\Phi(X), T))$. This is equivalent to minimizing the "Adjustment" part in equation 8.

4. Plug-in $h^*(X, T)$ to estimate $A\hat{T}E$:

$$\widehat{ATE}_{TMLE} = \frac{1}{N} \sum_{i=1}^{N} \underbrace{h^*(X_i, 1)}_{Y_i^*(1)} - \underbrace{h^*(X_i, 0)}_{Y_i^*(0)}$$

Targeted Regularization takes TMLE and adapts it for a neural network loss function. The main difference is that steps 1 and 2 above are done concurrently by Dragonnet, and that the loss functions

---

[9]For a deeper dive on targeted learning, we recommend (Benkeser and Chambaz, ????).

for the first three steps are combined into a single loss applied to the whole network at the end of each batch. It requires adding a single free parameter to the Dragonnet network for ϵ.

At a very intuitive level, Targeted Regularization is appealing because it introduces a loss function to TARNet that explicitly encourages the network to learn the mean of the treatment effect distribution, and not just the outcome distribution. The Targeted Regularization procedure proceeds as follows:

In each epoch:

1. (a) Use Dragonnet to predict $h(\Phi(X), T)$ and $\pi(\Phi(X), T)$.
   (b) Calculate the standard ML loss for the network using a hyperparameter α:

$$\underset{\Phi, \pi, h}{\arg\min} \; \underbrace{MSE(Y_i, h(\Phi(X_i), T_i))}_{\text{Outcome Loss}} + \alpha \underbrace{\mathrm{BCE}(T_i, \pi(\Phi(X_i), T_i))}_{\pi \text{ Loss}} + \lambda \underbrace{\mathcal{R}(h)}_{L_2}$$

2. (a) Compute $h^*(\Phi(X_i), T_i)$ as above,

$$\underbrace{h^*(\Phi(X_i), T_i)}_{Y^*} = \underbrace{h(\Phi(X_i), T_i)}_{\hat{Y}_i} + \underbrace{\left(\frac{T_i}{\pi(\Phi(X_i), 1)} - \frac{1 - T_i}{\pi(\Phi(X_i), 0)} \right)}_{\text{Treatment Modeling Adjustment}} \times \underbrace{\epsilon}_{\text{``nudge''}}$$

   (b) Calculate the targeted regularization loss: $MSE(Y, h^*(\Phi(X), T))$

3. Combine and minimize the losses from 1 and 2 using a hyperparameter β,

$$\underset{\Phi, h, \epsilon}{\arg\min} = \underbrace{MSE(Y, h(\Phi(X), T))}_{\text{Outcome Loss}} + \alpha \cdot \underbrace{\mathrm{BCE}(T, \pi(\Phi(X), T))}_{\pi \text{ Loss}} + \lambda \underbrace{\mathcal{R}(h)}_{L_2} + \beta \cdot \underbrace{MSE(Y, h^*(\Phi(X), T))}_{\text{Targeted Regularization Loss}}$$

Step 3 of Targeted Regularization is exactly equivalent to minimizing the EIC up to a constant β.

At the end of training, we can thus estimate the targeted regularization estimate of the ATE $\hat{ATE}_{TR}$ as in TMLE:

$$\hat{ATE}_{TR} = \frac{1}{N} \sum_{i=1}^{N} \underbrace{h^*(\Phi(X_i), 1)}_{Y_i^*(1)} - \underbrace{h^*(\Phi(X_i), 0)}_{Y_i^*(0)}$$

Compared to S-learners, T-learners, and TARNet, the Dragonnet algorithm is particularly attractive because of the statistical guarantees afforded by its semiparametric framework. It is doubly robust, unbiased, converges at a rate of $\frac{1}{\sqrt{n}}$, and the sampling distribution is asymptotically normal. Below we describe how to create assymptotically-valid confidence intervals for this estimator.

5   Confidence and Interpretation

In this section, we move from theory to practice, and treat best practices for building confidence intervals and interpreting heterogeneous treatment effects. Both of these topics are active areas of development, not only within the causal inference literature, but across machine learning research. Here we specifically focus on recommendations that can be easily implemented by analysts.

5.1   Asssesing Confidence

(Tutorial 4 [CO Open in Colab])

In this paper, we feature Dragonnet over other approaches because of its attractive statistical properties. Because the Targeted Regularization procedure in Dragonnet is essentially a variant of TMLE, an asymptotically valid standard error can be calculated as the sample corrected variance of the efficient influence curve $\sigma_{A\hat{T}E}$, where

$$\sigma_{AT\hat{E}_{TR}} = \sqrt{\frac{Var(EIC_{AT\hat{E}_{TR}})}{N}} \tag{9}$$

and,

$$Var(EIC_{AT\hat{E}_{TR}}) = Var[(\frac{T}{\pi(X,1)} - \frac{1-T}{\pi(X,0)})(Y - h^*(X,T)) + (h^*(X,1) - h^*(X,0)) - AT\hat{E}_{TR}] \tag{10}$$

((Van der Laan and Rose, 2011), pp. 96)

In Tutorial 5, we show how $\sigma_{A\hat{T}E}$ can be used to calculate a Wald confidence interval for Dragonnet. While not featured in this review, asymptotically valid conference intervals can also be calculated using RieszNet, a variant of Dragonnet introduced in Chernozhukov et al. (2022) that connects neural network estimation to the automatically debiased machine learning literature currently popular in causal econometrics (Chernozhukov et al., 2018, 2021).

5.2 Interpretation

(Tutorial 4 [CO Open in Colab])

A lack of interpretability has been a barrier to the adoption of machine learning methods like neural networks and random forests in social science settings. However, the literature on post-hoc interpretability techniques has matured considerably over the past five years, and several techniques for identifying important features/covariates such as permutation importance, LIME scores, SHAP scores, Individual Conditional Expectation plots etc... are in widespread usage today (Altmann et al., 2010; Goldstein et al., 2015; Lundberg and Lee, 2017; Ribeiro et al., 2016). For a broad and accessible treatment on interpretring machine learning models, see Molnar (2022).

Building on criteria used to evaluate other explainable AI methods, Crabbé et al. (2022) note four desirable properties of a feature importance technique for the interpretation of deep causal estimators: sensitivity, completeness, linearity, and implementation invariance (Sundararajan et al., 2017). A method that is 'sensitive' can distinguish between features that are simply predictive of the outcome, and those that actually influence CATE heterogeneity. A method that is 'complete' identifies all features that, together, explain all effect heterogeneity compared to a baseline. A 'linear' method is one where the feature importance scores additively describe the prediction. Lastly, the approach should be agnostic to both the model architecture (e.g., TARNet, Dragonnet) and different architectural hyper-parameterizations (i.e., invariant to implementation). Of the feature importance methods surveyed, they identify two that manifest all four of these qualities: SHAP scores, and integrated gradients.

SHAP (SHapley Additive exPlanations) scores have emerged as one of the most popular methods for evaluating machine learning models in recent years (Lundberg and Lee, 2017). SHAP is what is called a "local" interpretability method: it provides feature importance estimates for each individual datum. Theoretically, SHAP frames feature importance estimation as a cooperative (game-theoretic) game between covariates to predict a specific outcome. Under the hood, the algorithm exhaustively compares all possible "coalitions" of covariates and their ability to predict the outcome (win the game). Predictions from this powerset of coalitions are used to calculate the additive marginal contributions of each feature in prediction using Shapley values. The disadvantage of SHAP is that, even with

36

computational tricks, calculating scores for every unit can become computationally intractable in high dimensional datasets. SHAP scores are interpreted in comparison to a causal baseline of the ATE.

Because of the computational expense of SHAP scores, Crabbé et al. (2022) also recommend another local-interpretability method called "Integrated Gradients" (Sundararajan et al., 2017). Intuitively, this algorithm draws a straight-line, linear path in feature space between the target input (individual unit) and a baseline (i.e., a hypothetical unit who is exactly average on all covariates). A feature importance score can then be constructed by calculating the gradient in prediction error along this path with respect to the feature of interest. Note that SHAP scores can also be understood theoretically within the path framework. From this perspective, coalitions are paths in which each feature is turned on sequentially, and the SHAP score is the expectation across these paths. This interpretation leads to a gradient-based algorithm for calculating SHAP scores specifically for neural networks, which is also in the SHAP package. In practice, we recommend that analysts experiment with both integrated gradients and SHAP scores.

5.3 What's in the tutorials?

To move from theory to empirics, the online tutorials show how to implement many of the ideas presented throughout this primer. The tutorials are hosted in notebooks in the Google Colaboratory environment. When users open a Colab notebook, Google immediately provides a free virtual machine with standard Python machine learning packages available. This means that readers need not install anything on their own computers to experiment with these models. The tutorials are written in the Python programming language and provide examples in both Tensorflow2 and Pytorch, the two most popular deep learning frameworks. We note that both Tensorflow2 and Pytorch have implementations in R. However, we strongly recommend that readers interested in getting into deep learning work in Python, which has a much richer ecosystem of third-party packages for machine learning.

Currently there are five tutorials:

- [CO Open in Colab] Tutorial 1 introduces S-learners, and T-learners before TARNet as a way to get familiar with building custom Tensorflow models.

- [Open in Colab] Tutorial 2 focuses on causal inference metrics and hyperparameter optimization. Because we do not observe counterfactual outcomes, it's not obvious how to optimize supervised learning models for causal inference. This tutorial introduces some metrics for evaluating model performance. In the first part, you learn how to assess performance on these metrics in Tensorboard. In the second part, we hack Keras Tuner to do hyperparameter optimization for TARNet, and discuss considerations for training models as estimators rather than predictors.

- [Open in Colab] Tutorial 3 highlights the semi-parametric extension to TARNet featured in Shi et al. (2019). We add treatment modeling to our TARNet model, and build an augmented inverse propensity score estimator. We then briefly describe the algorithm for Targeted Maximum Likelihood Estimation to introduce and build a Dragonnet with Shi et al.'s Targeted Regularization.

- [Open in Colab] Tutorial 4 reimplements Dragonnet in Pytorch and shows how to calculate asymptotically-valid confidence intervals for the average treatment effect. We also interpret the features contributing to different heterogeneous CATEs using Integrated Gradients and SHAP scores. This tutorial is a good tutorial if you also just want to learn how to interpret SHAP scores, independent of the context of causal inference.

- [Open in Colab] Tutorial 5 features the Counterfactual Regression Network (CFRNet) and propensity-weighted CFRNet in Shalit et al. (2017); Johansson et al. (2018, 2020) (Appendix A.A). This approach relies on integral probability metrics to bound the counterfactual prediction loss and force the treated and control distributions closer together. The weighted variant adds adaptive propensity-based weights that provide a consistency guarantee, relax overlap assumptions, and ideally reduce bias.

6 Beyond Traditional Data: Text, Networks, Images, and Treatment over Time

As exciting as neural networks are for heterogeneous treatment effect estimation from quantitative data, a great promise of deep causal estimation is inference when treatments, confounders, and media-

tors are encoded in high-dimensional data (e.g., text, images, social networks, speech, and video) or are time-varying. This is a strong advantage of neural networks over other machine learning approaches, which do not generalize competitively to non-quantitative data. In these scenarios, multi-task objectives and tailored architectures can be used to learn representations that are simultaneously rich, capture information about causal quantities, and disentangle their relationships. Moreover, the inherent flexibility of neural networks means that, in many cases, the TARNet-style models presented above can serve as the foundations to inference on text and graphs with some architectural modifications, additional losses, and new identification assumptions.

This literature is rapidly evolving, so readers should treat this section of the primer as fundamentally prospective. To maintain accessibility, our primary goal here is to introduce readers to hypothetical scenarios where they might perform causal inference on text, network, or image data. Second, we selectively review contemporary, theoretically-motivated literature on deep causal estimation in these settings. The identification assumptions for different data types differ substantially, so we generally leave those to the interested reader. Finally, we briefly discuss approaches for dealing with time-varying confounding. We also take this section as an opportunity to introduce the Transformer or Graph Neural Network, an architecture now used in most contemporary deep learning models to learn from complex data (Box 8).

6.1   Causal Inference from Text

In recent years, an interdisciplinary community across both social science and computer science has coalesced around causal inference from text (see Keith et al. (2020) and Feder et al. (2021) for exhaustive reviews). Broadly speaking, texts may capture information about any causal quantity (treatments, outcomes, confounders, mediators) we might be interested in. For example, in an exit-polling experiment, analysts might want to measure toxicity (Y) in text responses to political prompts. In an observational study of e-mail response times (Y), analysts might want to measure the effects of the tone of the email (T). In this scenario, the analyst might also want to control for confounders like subject matter (X). Each of these scenarios presents distinct identification challenges (Feder et al., 2021). But in all cases, we can use low-dimensional representations of the high dimensional text to extract,

quantify, and disentangle relationships between nuanced qualities like tone and subject matter.

The ability of neural networks to automatically extract features makes them particularly suited for the last scenario when both treatment information and confounding covariates are encoded in text. In many cases, we may not have explicitly identified, quantified, or labeled all of the confounders in text (e.g., subject matter and tone of emails), but we would still like to control for them. Pryzant et al. (2021),Veitch et al. (2020), and Gui and Veitch (2022) address this problem by prepending Transformer-layers (Box 8) for reading text to the beginning of TARNet or Dragonnet. Veitch et al. (2020) demonstrate the viability of this approach on a Science of Science question testing the causal effect of equations on getting papers accepted to computer science conferences. Pryzant et al. (2021); Gui and Veitch (2022) explore the more complicated scenario not in which the treatment is explicitly known (e.g., equations in papers, gender of authors), but is instead externally perceived upon reading (e.g., politeness/rudeness of an email or toxicity of a social media post). In these models, an additional loss function is also added for learning text representations concurrently with the causal inference losses discussed above.

Box 8: Graph Neural Networks and Transformers

Graph neural networks (GNNs) are the current state-of-the-art approach for creating representations for nodes in graphs. Compared to previous approaches that relied on "shallow" embeddings based only on a node's local context (e.g., random walks to nearby nodes), GNNs are attractive because their node representations are aggregated from the structural position and covariates of all nodes n degrees away from the target node, where n is the number of graph neural network layers.

The most intuitive understanding of how graph neural networks work is as a message passing system (Gilmer et al., 2017). We use one of the first GNN papers, the Graph Convolutional Network as an example (Kipf and Welling, 2017). In this interpretation, each node has a message that it passes to it's neighbors through a graph convolution operation. In the first layer of a GNN this message would consist of the node's covariates/features. In consecutive layers of the network, these messages are actually representations of the node produced by the previous layer. During graph convolution, each node multiplies incoming messages by it's own set of weights and combines these weighted inputs using an aggregation function (e.g., summation). By the n-th GNN layer, these messages will contain structure and covariate information from all nodes n degrees away. For interested readers, there is also a spectral interpretation of this process. Typically GNNs are trained to produce representations of graphs by predicting the probability that two nodes are linked in the network, and then used for something else. One variant of the GNN uses an "attention" mechanism to vary the extent that nodes value messages from different neighbors (the graph attention network or GAT) (Veličković et al., 2018).

40

As of 2023, **Transformers** are the hegemonic architecture used in natural language processing. After their introduction in 2017, these models improved performance on many high-profile NLP tasks across the board. Several enterprise-scale transformers have been featured in the media for their impressive performance in text generation and question answering (e.g. OpenAI's GPT-3 and ChatGPT, Google's Bard). Smaller models in broad use are based on the BERT architecture (Devlin et al., 2019).

The connection between GNNs, and specifically GATs, is the focus on attention mechanisms. From this perspective, words in sentences are akin to nodes in networks, with their relative positions to each other being analogous to their structural positions in the graph. Transformers improved on previous sequential approaches to text analysis (i.e. recurrent neural networks) by having each word (or representation of a word) receive messages from not just adjacent words, but all words heterogeneously. Attention mechanisms throughout the architecture allow each layer of a transformer to attend to words or aggregated representation mechanisms heterogeneously. Architectures such as BERT or GPT stack transformer layers to create models with hundreds of millions of parameters. These models are expensive to train, both computationally and with respect to data, so they are often pretrained on large datasets and then "fine-tuned" (lightly re-trained) with smaller datasets for specific tasks or to align with certain goals.

6.2 Causal Inference from Networks

A smaller literature has leveraged relational data for causal inference in two distinct scenarios. In the first traditional selection on observable settings, we wish to control for information about unobserved confounding inferable from homophilous ties. For example, age or gender might be unmeasured in our data, but we might expect people to develop friendship ties with those of the same gender identity or age cohort.

This scenario suggests estimation strategies similar to those when confounders are encoded in text. Much like Transformer layers can be prepended to TARNet-style estimators to learn from text, graph neural networks (an analog of the Transformer) can be preprended to learn from graphs. Guo et al. (2020) provides a first pass at this problem by adding GNN layers to CFRNet Shalit et al. (2017). Veitch et al. (2019) instead adapt Dragonnet in a semi-parametric framework to allow for consistent estimates of the treatment and outcome, assuming the network representation encodes significant information about confounders.

The second, more challenging scenario is estimating the causal effect of social influence on outcomes from observational data. For example, Cristali and Veitch (2022) introduce the problem of measuring

the effects of vaccination (T) on peer vaccination choice (Y). This is a hard problem because a) SUTVA is a fundamental assumption of all causal inference frameworks and b) it is hard to disentangle whether changes in the outcome result from the treatment via peer effects (e.g, person A pressuring person B to vaccinate), or from homophily (e.g., person A and person B having similar political leanings). In other words, contagion and homophily are generically confounded (Shalizi and Thomas, 2011). McFowland and Shalizi (2021) are the first to tackle this problem by making strong parametric assumptions about the generation of network ties and the outcome model. Cristali and Veitch (2022) instead propose an approach using neural network-learned representations of the graph.

6.3 Causal Inference from Images

While ideas from causal inference have been leveraged extensively to improve image classification, to our knowledge there are no papers that explore causal inference where treatments, confounders, mediators, or predictors are encoded in images.[10] That being said, some scenarios proposed for causal text analysis should apply here as well. For example, consider the conjoint experiment by Todorov et al. (2005) where both the treatment (e.g. incumbency of a politician) and potential latent confounders (e.g., party, age, gender, race) are encoded in an image. In this setting, a TARNet-like model adapted to learn and condition on image representations could improve treatment effect estimation by controlling for confounders such as the politician's age. Causal inference on images is an area ripe for exploration, and we hope to see more work here in the future.

6.4 Causal Inference from Time-varying Data

One natural extension of deep causal estimation is to scenarios where treatments are administered over time and confounding may be time-varying. While "g-methods" developed by Robins et al. for estimating effects with time-varying treatments and confounding have existed for decades, the statistical assumptions encoded in these models are quite strong (Robins, 1994; Robins et al., 2000, 2009). Due to their reliance on generalized linear models to define the "structural" component, they

---

[10]Jesson et al. (2021) introduce a simulation where the MNIST digit dataset serves as covariates X as toy example of high-dimensional confounding, but not a possible application.

assume that the outcome is a linear function of all covariates and treatment. Second, for identification, they make strong assumptions about which previous timesteps confound the current one. Third, they require different coefficients to be estimated at each time steps. Transformers (Box 8) and recurrent neural networks, a simpler model for sequential data (Appendix A.C), should be able to capture long-term dependencies and non-linearities in ways that marginal structural models and g-computation cannot.

Several papers have begun to explore these possibilities in the context of personalized medicine. Lim et al. (2018) build a marginal structural model using a recurrent neural network, and Bica et al. (2020a) extend this framework with an additional loss to more explicitly deal with time varying confounding by forcing the model to "unlearn" information about the previous time steps. Melnychuk et al. (2022) go one step further by adapting Bica et al. (2020a)'s approach with a transformer. Inspired by longitudinal targeted maximum likelihood, Frauen et al. (2022) add a semi-parametric targeting layer to their RNN to create a g-computation algorithm that is doubly robust and asymptotically efficient. Li et al. (2021) instead propose an RNN framework for g-computation that allows for dynamic treatment regimes. All of these papers use simulations of tumor growth dynamics, naturalistic simulations based on vital signs from intensive care unit visits, or factual datasets exploring treatment response to physical therapy for back pain.

7   Conclusion: Deep Causal Estimation in Context

In this primer we introduce social scientists to the emerging machine learning literature on deep learning for causal inference. To set the stage, we first provide both an intuitive introduction to fundamental deep learning concepts like representation and multi-task learning, as well as practical guidelines for training neural networks. In the main body of the article, we show how ML researchers have adapted core treatment and outcome modeling strategies to leverage the particular strengths of neural networks for heterogeneous treatment effect estimation. We follow with a discussion on inference (e.g., model selection, confidence intervals, interpretation), and closed with a prospective look at algorithms for inference from text, social networks, images, and time varying data.

Deep learning is not the only potential tool for heterogeneous treatment effect inference, and there are robust literatures exploring the usage of other methods in both the econometrics and biostatistics communities (Van der Laan and Rose, 2011; Chernozhukov et al., 2018; Wager and Athey, 2018). While these literatures are certainly more mature, below we discuss reasons why we think the use gap between neural networks and other machine learning methods will continue to narrow, a change that we must prepare for.

First, neural networks are better at modeling non-linear heterogeneity (e.g., in treatment responses) than other machine learning methods. In extensive simulations, Curth et al. (2021) found that when the data-generating process for treatment heterogeneity includes exponential relationships, neural networks outperformed random forests, but tree-based methods are robust when the data-generating process is built on linear functions. Neural networks were also consistently better at predicting outlier treatment effects than forests. These differences result from how the two methods model functions. While neural networks can approximate any continuous function with enough neurons, random forests must build non-linear or non-orthogonal decision boundaries using piecewise functions and average predictions. Consistent with these differences, Curth et al. (2021) also find that neural networks do better when variables are constructed as continuous covariates, and vice versa when they are dichotomized.

From a statistical perspective, the rise of semi-parametric and double machine learning frameworks has also narrowed the gap between neural networks and other types of machine learning in terms of theoretical guarantees. For example, the TMLE-inspired Dragonnet algorithm featured here is unbiased, plausibly consistent, and converges to the target estimand at a fast rate of $\frac{1}{\sqrt{n}}$. The closely-related Riezsnet double machine learning model (not featured) boasts similar guarantees (Chernozhukov et al., 2022). Beyond these algorithms, there is a growing adjacent literature of model-agnostic plug-in learners (e.g., X-learner, R-learner) that can leverage the strengths of neural networks (Nie and Wager, 2021; Künzel et al., 2019).

Third, folk beliefs about the data-hungriness and uninterpretability of neural networks are overstated. Neural networks are data-hungry when over-parameterized or learning from high-dimensional data like images, but we show in the tutorials that modest-sized, well-regularized neural networks

can successfully infer heterogeneous treatment effects in a naturalistic simulation of quantitative data with less than 800 units. In Section 5, we also highlight the considerable progress in machine learning interpretability over the past five years, much of which has been on model-agnostic approaches that benefit all black-box algorithms equally.[11]

In our opinion, the most pressing limitation of current deep learning approaches is the difficulty of optimizing neural networks. Theoretically, this stems from a) the complexity of the loss functions which are often non-convex, and b) the ease of over-parameterizing these models to fit these functions. If neural networks are to be used as statistical estimators, statistical guarantees must be backed by optimization guarantees and/or more rigorous methods for model selection. Outside of statistical estimation, this limitation has largely been addressed through empirical testing on test data and strategic model selection. Within the statistical estimation context, this gap will likely need to be addressed by simulation-based sensitivity analyses and, in the short term, comparisons to other model families.

Moreover, there has been a lack of mature tools and empirical applications of these models. A major goal of this primer, and the tutorials in particular, is to synthesize the theoretical literature, practical training and interpretation guidelines, and annotated code so that social scientists in one place can start using these models. Deep learning frameworks like Tensorflow and Pytorch are becoming more accessible every year, but we note that canned Python packages like Uber's causalML exist for interested readers who just want to experiment with a few of these models (Chen et al., 2020).

Despite current limitations, we believe the future of causal estimation runs through deep learning. As causal inference ventures into new settings, the flexibility of neural networks will become essential for learning from text, graph, image, video, and speech data. For time-varying settings, we believe the ability of neural networks to model non-linearities and long-range temporal dependencies will ultimately lead to solutions with net weaker assumptions than current approaches. Overall, we are optimistic and excited to see where deep causal estimation heads over the next few years.

---

[11] Critics often point to out-of-bag feature importances as a particular strength of random forests, but this approach has been shown to be less accurate than model-agnostic permutation importances anyways (Altmann et al., 2010).

References

Alaa, Ahmed and Mihaela Van Der Schaar. 2019. "Validating Causal Inference Models via Influence Functions." In *International Conference on Machine Learning*, volume 36, pp. 191–201. Association for Computing Machinery.

Altmann, André, Laura Toloşi, Oliver Sander, and Thomas Lengauer. 2010. "Permutation importance: a corrected feature importance measure." *Bioinformatics* 26:1340–1347.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. 2017. "Wasserstein Generative Adversarial Networks." In *International Conference on Machine Learning*, volume 34, pp. 214–223. Association for Computing Machinery.

Atan, Onur, James Jordon, and Mihaela Van Der Schaar. 2018. "Deep-Treat: Learning Optimal Personalized Treatments from Observational Data Using Neural Networks." In *Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, volume 32, p. 2071–2078. Association for the Advancement of Artificial Intelligence.

Athey, Susan and Guido Imbens. 2016. "Recursive partitioning for heterogeneous causal effects." *Proceedings of the National Academy of Sciences* 113:7353–7360.

Austin, Peter C. 2011. "An Introduction to Propensity Score Methods for Reducing the Effects of Confounding in Observational Studies." *Multivariate Behavioral Research* 46:399–424.

Bengio, Yoshua. 2013. "Deep Learning of Representations: Looking Forward." In *International Conference on Statistical Language and Speech Processing*, pp. 1–37. Association for Computing Machinery.

Benkeser, D, M Carone, M J Van Der Laan, and P B Gilbert. 2017. "Doubly robust nonparametric inference on the average treatment effect." *Biometrika* 104:863–880.

Benkeser, David and Antoine Chambaz. ???? *A Ride in Targeted Learning Territory*.

Bica, Ioana, Ahmed M Alaa, James Jordon, and Mihaela van der Schaar. 2020a. "Estimating Counterfactual Treatment Outcomes Over Time through Adversarially Balanced Representations." In

International Conference on Learning Representations, volume 37. Association for Computing Machinery.

Bica, Ioana, James Jordon, and Mihaela van der Schaar. 2020b. "Estimating the Effects of Continuous-valued Interventions using Generative Adversarial Networks." In *Neural Information Processing Systems*, volume 33, pp. 16434–16445.

Bodnar, Lisa M, Abigail R Cartus, Edward H Kennedy, Sharon I Kirkpatrick, Sara M Parisi, Katherine P Himes, Corette B Parker, William A Grobman, Hyagriv N Simhan, Robert M Silver, Deborah A Wing, Samuel Perry, and Ashley I Naimi. 2022. "Use of a Doubly Robust Machine-Learning–Based Approach to Evaluate Body Mass Index as a Modifier of the Association Between Fruit and Vegetable Intake and Preeclampsia." *American Journal of Epidemiology* 191:1396–1406.

Brand, Jennie E, Bernard Koch, and Jiahui Xu. 2020. "Machine Learning." In *Sage Research Methods Foundations*. SAGE.

Chen, Huigang, Totte Harinen, Jeong-Yoon Lee, Mike Yung, and Zhenyu Zhao. 2020. "CausalML: Python Package for Causal Machine Learning."

Chernozhukov, Victor, 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:C1–C68.

Chernozhukov, Victor, Whitney Newey, Victor M Quintas-Martinez, and Vasilis Syrgkanis. 2022. "Riesznet and forestriesz: Automatic debiased machine learning with neural nets and random forests." In *International Conference on Machine Learning*, pp. 3901–3914. PMLR.

Chernozhukov, Victor, Whitney K Newey, Victor Quintas-Martinez, and Vasilis Syrgkanis. 2021. "Automatic debiased machine learning via neural nets for generalized linear regression." *arXiv preprint arXiv:2104.14737* .

Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. "Learning Phrase Representations using RNN Encoder–Decoder

for Statistical Machine Translation." In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. Association for Computational Linguistics.

Crabbé, Jonathan, Alicia Curth, Ioana Bica, and Mihaela van der Schaar. 2022. "Benchmarking heterogeneous treatment effect models through the lens of interpretability." *arXiv preprint arXiv:2206.08363* .

Cristali, Irina and Victor Veitch. 2022. "Using Embeddings for Causal Estimation of Peer Influence in Social Networks." *ArXiv* abs/2205.08033.

Curth, Alicia, David Svensson, Jim Weatherall, and Mihaela van der Schaar. 2021. "Really Doing Great at Estimating CATE? A Critical Look at ML Benchmarking Practices in Treatment Effect Estimation." In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, edited by J. Vanschoren and S. Yeung, volume 1.

Cuturi, Marco. 2013. "Sinkhorn Distances: Lightspeed Computation of Optimal Transport." In *Neural Information Processing Systems*, volume 27, pp. 2292–2300. Curran Associates, Inc.

Cybenko, George. 1989. "Approximation by superpositions of a sigmoidal function." *Mathematics of Control, Signals and Systems* 5:455.

Daza, Daniel. 2019. "Approximating Wasserstein Distances with PyTorch." `https://dfdazac.github.io/sinkhorn.html`. Last accessed 2019-08-01.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186. Association for Computational Linguistics.

Du, Xin, Lei Sun, Wouter Duivesteijn, Alexander Nikolaev, and Mykola Pechenizkiy. 2021. "Adversarial Balancing-based Representation Learning for Causal Effect Inference with Observational Data." *Data Mining and Knowledge Discovery* 35:1713–1738.

Farrell, Max H, Tengyuan Liang, and Sanjog Misra. 2021. "Deep Neural Networks for Estimation and Inference." *Econometrica* 89:181–213.

Feder, Amir, Katherine A Keith, Emaad Manzoor, Reid Pryzant, Dhanya Sridhar, Zach Wood-Doughty, Jacob Eisenstein, Justin Grimmer, Roi Reichart, Margaret E Roberts, et al. 2021. "Causal Inference in Natural Language Processing: Estimation, Prediction, Interpretation and Beyond." *arXiv preprint arXiv:2109.00725* .

Fisher, Aaron and Edward H Kennedy. 2021. "Visually Communicating and Teaching Intuition for Influence Functions." *The American Statistician* 75:162–172.

Frauen, Dennis, Tobias Hatt, Valentyn Melnychuk, and Stefan Feuerriegel. 2022. "Estimating average causal effects from patient trajectories." *arXiv preprint arXiv:2203.01228* .

Gilmer, Justin, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. "Neural message passing for quantum chemistry." In *International Conference on Machine Learning*, volume 34, pp. 1263–1272. Association for Computing Machinery.

Glynn, Adam N and Kevin M Quinn. 2010. "An introduction to the Augmented Inverse Propensity Weighted Estimator." *Political Analysis* 18:36–56.

Goldstein, Alex, Adam Kapelner, Justin Bleich, and Emil Pitkin. 2015. "Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation." *Journal of Computational and Graphical Statistics* 24:44–65.

Goldszmidt, Moisés and Judea Pearl. 1996. "Qualitative probabilities for default reasoning, belief revision, and causal modeling." *Artificial Intelligence* 84:57–112.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative Adversarial Nets." In *Neural Information Processing Systems*, volume 27, pp. 2672–2680. Association for Computing Machinery.

Gretton, Arthur, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. "A Kernel Two-sample Test." *Journal of Machine Learning Research* 13:723–773.

Gui, Lin and Victor Veitch. 2022. "Causal Estimation for Text Data with (Apparent) Overlap Violations." *arXiv preprint arXiv:2210.00079* .

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. "Improved Training of Wasserstein GANs." In *International Conference on Neural Information Processing Systems*, volume 31, p. 5769–5779. Curran Associates Inc.

Guo, Ruocheng, Jundong Li, and Huan Liu. 2020. "Counterfactual Evaluation of Treatment Assignment Functions with Networked Observational Data." In *SIAM International Conference on Data Mining*, pp. 271–279. Society for Industrial and Applied Mathematics.

Hastie, T., R. Tibshirani, and J.H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer.

Heck, Katherine E, Paula Braveman, Catherine Cubbin, Gilberto F Chávez, and John L Kiely. 2006. "Socioeconomic status and breastfeeding initiation among California mothers." *Public health reports* 121:51–59.

Hernán, Miguel A. and James M. Robins. 2020. *Causal Inference: What If. 2020*. Chapman & Hall.

Hill, Jennifer L. 2011. "Bayesian Nonparametric Modeling for Causal Inference." *Journal of Computational and Graphical Statistics* 20:217–240.

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9:1735–1780.

Holland, Paul W. 1986. "Statistics and Causal Inference." *Journal of the American statistical Association* 81:945–960.

Huszar, Ferenc. 2015. "Another Favourite Machine Learning Paper: Adversarial Networks vs Kernel Scoring Rules." Last accessed 2019-08-01.

Imbens, Guido W and Donald B Rubin. 2015. *Causal Inference in Statistics, Social, and Biomedical Sciences*. Cambridge University Press.

Ioffe, Sergey and Christian Szegedy. 2015. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In *International Conference on Machine Learning*, volume 32, pp. 448–456. Association for Computing Machinery.

Jesson, Andrew, Sören Mindermann, Yarin Gal, and Uri Shalit. 2021. "Quantifying Ignorance in Individual-Level Causal-Effect Estimates under Hidden Confounding." *arXiv preprint arXiv:2103.04850* .

Johansson, Fredrik and Max Shen. 2018. "Causal Inference & Deep Learning." https://github.com/maxwshen/iap-cidl. MIT IAP.

Johansson, Fredrik D, Nathan Kallus, Uri Shalit, and David Sontag. 2018. "Learning Weighted Representations for Generalization Across Designs." *Unpublished* .

Johansson, Fredrik D., Uri Shalit, Nathan Kallus, and David A. Sontag. 2020. "Generalization Bounds and Representation Learning for Estimation of Potential Outcomes and Causal Effects." *arXiv* abs/2001.07426.

Johansson, Fredrik D, Uri Shalit, and David Sontag. 2016. "Learning Representations for Counterfactual Inference." In *International Conference on Machine Learning*, volume 48. Association for Computing Machinery.

Joo, Jungseock, Francis F Steen, and Song-Chun Zhu. 2015. "Automated Facial Trait Judgment and Election Outcome Prediction: Social Dimensions of Face." In *International Conference on Computer Vision*, pp. 3712–3720. IEEE.

Kallus, Nathan. 2020. "Generalized Optimal Matching Methods for Causal Inference." *Journal of Machine Learning Research* 21:62–1.

Keith, Katherine, David Jensen, and Brendan O'Connor. 2020. "Text and Causal Inference: A Review of Using Text to Remove Confounding from Causal Estimates." In *Annual Meeting of the Association*

for Computational Linguistics, volume 58, pp. 5332–5344, Online. Association for Computational Linguistics.

Kennedy, Edward H. 2016. "Semiparametric Theory and Empirical Processes in Causal Inference." In *Statistical Causal Inferences and their Applications in Public Health Research*, pp. 141–167. Springer.

Kennedy, Edward H. 2020. "Towards optimal doubly robust estimation of heterogeneous causal effects."

Kingma, Diederik P. and Jimmy Ba. 2015. "Adam: A Method for Stochastic Optimization." In *International Conference on Learning Representations*, volume 3. OpenReview.

Kipf, Thomas N and Max Welling. 2017. "Semi-supervised classification with graph convolutional networks." *International Conference on Learning Representations* 5.

Kramer, Michael S, Frances Aboud, Elena Mironova, Irina Vanilovich, Robert W Platt, Lidia Matush, Sergei Igumnov, Eric Fombonne, Natalia Bogdanovich, Thierry Ducruet, et al. 2008. "Breastfeeding and child cognitive development: new evidence from a large randomized trial." *Archives of general psychiatry* 65:578–584.

Künzel, Sören R, 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* 116:4156–4165.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep learning." *Nature* 521:436.

Li, Rui, Stephanie Hu, Mingyu Lu, Yuria Utsumi, Prithwish Chakraborty, Daby M. Sow, Piyush Madan, Jun Li, Mohamed Ghalwash, Zach Shahn, and Li-wei Lehman. 2021. "G-Net: a Recurrent Network Approach to G-Computation for Counterfactual Prediction Under a Dynamic Treatment Regime." In *Proceedings of Machine Learning for Health*, edited by Subhrajit Roy, Stephen Pfohl, Emma Rocheteau, Girmaw Abebe Tadesse, Luis Oala, Fabian Falck, Yuyin Zhou, Liyue Shen, Ghada Zamzmi, Purity Mugambi, Ayah Zirikly, Matthew B. A. McDermott, and Emily Alsentzer, volume 158 of *Proceedings of Machine Learning Research*, pp. 282–299. PMLR.

Lim, Bryan, Ahmed M Alaa, and Mihaela van der Schaar. 2018. "Forecasting Treatment Responses Over Time Using Recurrent Marginal Structural Networks." In *Neural Information Processing Systems*, volume 18, pp. 7483–7493. Curran Associates Inc.

Lundberg, Scott M and Su-In Lee. 2017. "A Unified Approach to Interpreting Model Predictions." In *Advances in Neural Information Processing Systems*, edited by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, volume 30. Curran Associates, Inc.

McFowland, Edward and Cosma Rohilla Shalizi. 2021. "Estimating Causal Peer Influence in Homophilous Social Networks by Inferring Latent Locations." *Journal of the American Statistical Association* 0:1–12.

Melnychuk, Valentyn, Dennis Frauen, and Stefan Feuerriegel. 2022. "Causal Transformer for Estimating Counterfactual Outcomes." In *Proceedings of the 39th International Conference on Machine Learning*, edited by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, volume 162 of *Proceedings of Machine Learning Research*, pp. 15293–15329. PMLR.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Distributed Representations of Words and Phrases and Their Compositionality." In *Neural Information Processing Systems*, volume 26, p. 3111–3119. Curran Associates Inc.

Molnar, C. 2022. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Christoph Molnar.

Müller, Alfred. 1997. "Integral Probability Metrics and Their Generating Classes of Functions." *Advances in Applied Probability* 29:429–443.

Nagpal, Chirag, Dennis Wei, Bhanukiran Vinzamuri, Monica Shekhar, Sara E. Berger, Subhro Das, and Kush R. Varshney. 2020. "Interpretable Subgroup Discovery in Treatment Effect Estimation with Application to Opioid Prescribing Guidelines." In *Conference on Health, Inference, and Learning*, p. 19–29. Association for Computing Machinery.

Naimi, Ashley I and Laura B Balzer. 2018. "Stacked generalization: an introduction to super learning." *European journal of epidemiology* 33:459–464.

Nie, Xinkun and Stefan Wager. 2021. "Quasi-oracle Estimation of Heterogeneous Treatment Effects." *Biometrika* 108:299–319.

Parikh, Harsh, Carlos Varjao, Louise Xu, and Eric Tchetgen Tchetgen. 2022. "Validating Causal Inference Methods." In *Proceedings of the 39th International Conference on Machine Learning*, edited by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, volume 162 of *Proceedings of Machine Learning Research*, pp. 17346–17358. PMLR.

Pearl, Judea. 2009. *Causality*. Cambridge University Press.

Pryzant, Reid, Dallas Card, Dan Jurafsky, Victor Veitch, and Dhanya Sridhar. 2021. "Causal Effects of Linguistic Properties." In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4095–4109. Association for Computational Linguistics.

Ramey, Craig T, Donna M Bryant, Barbara H Wasik, Joseph J Sparling, Kaye H Fendt, and Lisa M La Vange. 1992. "Infant Health and Development Program for low birth weight, premature infants: Program elements, family participation, and child intelligence." *Pediatrics* 89:454–465.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. "" Why should i trust you?" Explaining the predictions of any classifier." In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.

Roberts, Daniel A., Sho Yaida, and Boris Hanin. 2022. *The Principles of Deep Learning Theory*. Cambridge University Press. https://deeplearningtheory.com.

Robins, James. 1986. "A New Approach to Causal Inference in Mortality Studies with a Sustained Exposure Period—Application to Control of the Healthy Worker Survivor Effect." *Mathematical Modelling* 7:1393–1512.

Robins, James. 1987. "A Graphical Approach to the Identification and Estimation of Causal Parameters in Mortality Studies with Sustained Exposure Periods." *Journal of Chronic Diseases* 40:139S–161S.

Robins, James M. 1994. "Correcting for non-compliance in randomized trials using structural nested mean models." *Communications in Statistics-Theory and Methods* 23:2379–2412.

Robins, James M, Miguel Angel Hernan, and Babette Brumback. 2000. "Marginal Structural Models and Causal Inference in Epidemiology." *Epidemiology* .

Robins, James M, Miguel A Hernán, G Fitzmaurice, M Davidian, G Verbeke, and G Molenberghs. 2009. "Longitudinal Data Analysis." *Handbooks of Modern Statistical Methods* pp. 553–599.

Rosenbaum, Paul R and Donald B Rubin. 1983. "The Central Role of the Propensity Score in Observational Studies for Causal Effects." *Biometrika* 70:41–55.

Rubin, Donald B. 1974. "Estimating Causal Effects of Treatments in Randomized and Non-randomized Studies." *Journal of Educational Psychology* 66:688.

Schnitzer, Mireille E, Judith J Lok, and Susan Gruber. 2016. "Variable selection for confounder control, flexible modeling and collaborative targeted minimum loss-based estimation in causal inference." *The international journal of biostatistics* 12:97–115.

Schwab, Patrick, Lorenz Linhardt, and Walter Karlen. 2018. "Perfect Match: A Simple Method for Learning Representations For Counterfactual Inference With Neural Networks." *arXiv:1810.07406v1* .

Shalit, Uri, Fredrik D Johansson, and David Sontag. 2017. "Estimating Individual Treatment Effect : Generalization Bounds and Algorithms." In *International Conference on Machine Learning*. Association for Computing Machinery.

Shalizi, Cosma Rohilla and Andrew C. Thomas. 2011. "Homophily and Contagion Are Generically Confounded in Observational Social Network Studies." *Sociological Methods & Research* 40:211–239.

Shi, Claudia, David Blei, and Victor Veitch. 2019. "Adapting Neural Networks for the Estimation of Treatment Effects." *Nneural Information Processing Systems* 32.

Snoek, Jasper, Hugo Larochelle, and Ryan P Adams. 2012. "Practical Bayesian Optimization of Machine Learning Algorithms." In *Advances in Neural Information Processing Systems*, edited by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, volume 25. Curran Associates, Inc.

Srivastava, Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research* 15:1929–1958.

Stock, Michiel. 2017. "Notes on optimal transport." https://michielstock.github.io/OptimalTransport/. Last accessed 2019-08-01.

Stuart, Elizabeth A. 2010. "Matching Methods for Causal Inference: A Review and a Look Forward." *Statistical Science* 25:1.

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan. 2017. "Axiomatic attribution for deep networks." In *International conference on machine learning*, pp. 3319–3328. PMLR.

Todorov, Alexander, Anesu N Mandisodza, Amir Goren, and Crystal C Hall. 2005. "Inferences of Competence from Faces Predict Election Outcomes." *Science* 308:1623–1626.

Van der Laan, Mark J and Sherri Rose. 2011. *Targeted Learning: Causal Inference for Observational and Experimental Data*. Springer Science & Business Media.

Veitch, Victor, Dhanya Sridhar, and David Blei. 2020. "Adapting Text Embeddings for Causal Inference." In *Conference on Uncertainty in Artificial Intelligence*, pp. 919–928. Association for Uncertainty in Artificial Intelligence.

Veitch, Victor, Yixin Wang, and David Blei. 2019. "Using Embeddings to Correct for Unobserved Confounding in Networks." *Neural Information Processing Systems* 32.

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. "Graph Attention Networks." In *International Conference for Learning Representations*, volume 6. OpenReview.

Wager, Stefan and Susan Athey. 2018. "Estimation and Inference of Heterogeneous Treatment Effects using Random Forests." *Journal of the American Statistical Association* 113:1228–1242.

Yao, Liuyi, Sheng Li, Yaliang Li, Mengdi Huai, Jing Gao, and Aidong Zhang. 2018. "Representation Learning for Treatment Effect Estimation from Observational Data." In *Neural Information Processing Systems*. Curran Associates, Inc.

Yoon, Jinsung, James Jordon, and Mihaela Van Der Schaar. 2018. "GANITE: Estimation of Individualized Treatment Effects using Generative Adversarial Nets." In *International Conference on Learning Representations*. OpenReview.

Zhang, Yao, Alexis Bellot, and Mihaela Schaar. 2020. "Learning Overlapping Representations for the Estimation of Individualized Treatment Effects." In *International Conference on Artificial Intelligence and Statistics*, pp. 1005–1014. Association for Artificial Intelligence and Statistics.

Zivich, Paul N and Alexander Breskin. 2021. "Machine learning for causal inference: on the use of cross-fit estimators." *Epidemiology (Cambridge, Mass.)* 32:393.

Appendix A   Balancing Using Integral Probability Metrics

A.1   Wasserstein Distance

Following (Stock, 2017; Daza, 2019), suppose we have two discrete distributions (treated and control) with marginal densities $p(x)$ and $q(x)$ captured as vectors t and c, with dimensions n and m respectively. To compute the Wasserstein distance, we must define a "mapping matrix" P that defines the mapping of "earth" in $p(x)$ to corresponding piles in $q(x)$. Let $\mathbf{U}(t, c)$ be the set of positive, $n \times m$ mapping

matrices where the sum of the rows is t and the sum of the columns is c.

$$\mathbf{U}(t,c) = P \subset \mathbb{R}_{>0}^{nxm} | P \cdot 1_m = \mathbf{t}, P^T \cdot 1_n = \mathbf{c} \tag{11}$$

In words, this matrix maps the probability mass from points in the support of $p(x)$ (i.e, the elements of t) to points in the support of $q(x)$ (the elements of c) (note that the mapping need not be one-to-one). We also have a "cost" matrix C that describes the cost of applying P (i.e. the cost of shoveling dirt according to the map described in P). The cost matrix can be computed using a norm ℓ (most commonly ℓ^2) between the points in t being mapped to c in the mapping matrix P. Finally, the ℓ-norm Wasserstein distance dW_ℓ can be defined as

$$dW_\ell = min_{P \subset (t,c)} \sum_{i,j} P_{ij} C_{ij} \tag{12}$$

In other words, the Wasserstein distance is the smallest Frobenius inner product of a mapping matrix P that fits the above constraints, and its associated cost matrix C. Although this problem can be solved via linear programming, the Wasserstein distance is often implemented in a different form that works with continuous distributions and can be optimized by gradient descent (Arjovsky et al., 2017; Gulrajani et al., 2017). There is also a variant of the Wasserstein distance that imposes an entropy-based regularization on the coupling matrix to make it smoother or sparser called the Sinkhorn distance (Cuturi, 2013).

A.2 Extending Representation Balancing with IPMs

Deep Dive: CFRNet (Shalit et al. (2017); Johansson et al. (2018, 2020))

Beyond receiving outcome modeling gradients for both potential outcomes, the authors have subsequently extended TARNet with additional losses that explicitly encourage balancing by minimizing a statistical distance between the two covariate distributions in representation space. These distances are called integral probability metrics (Müller, 1997).[12] Johansson et al. (2016); Shalit et al. (2017);

---

[12]Zhang et al. (2020) criticize the usage of IPMs because they make no restrictions on the moments of the transformed distributions. Thus while the covariate distributions may have a high percentage of overlap in representation space, this

58

Johansson et al. (2018) propose two possible IPMs, the Wasserstein distance and the maximum mean discrepancy distance (MMD) for use in these architectures.

The Wasserstein or "Earth Mover's" distance fits an interpretable "map" (i.e. a matrix) showing how to efficiently convert from one probability mass distribution to another. The Wasserstein distance is most easily understood as an optimal transport problem (i.e., a scenario where we want to transport one distribution to another at minimum cost). The nickname "Earth mover's distance" comes from the metaphor of shoveling dirt to terraform one landscape into another. In the idealized case in which one distribution can be perfectly transformed into another, the Wasserstein map corresponds exactly to a perfect one-to-one matching on covariates strategy (Kallus, 2020).

The MMD is the normed distance between the means of two distributions, after a kernel function ϕ has transformed them into a high-dimensional space called a reproducing kernel Hibbert Space (RKHS) (Gretton et al., 2012). The MMD with an L^2 norm in RKHS $\mathcal{H}$ can be specified as:

$$MMD(P, Q) = ||\mathbb{E}_{X \sim P} \phi(X) - \mathbb{E}_{X \sim Q} \phi(X)||_{\mathcal{H}}^2 \tag{13}$$

The metric is built on the idea that there is no function that would have differing Expected Values for P and Q in this high-dimensional space if P and Q are the same distribution (Huszar, 2015). The MMD is inexpensive to calculate using the "kernel trick" where the inner product between two points can be calculated in the RKHS without first transforming each point into the RKHS.[13]

When an IPM loss is applied to the representation layers in TARNet, the authors call the resulting network "CounterFactual Regression Network" (CFRNet) (Fig. 6A) (Shalit et al., 2017). The loss function for this network is

$$\min_{h, \Phi, IPM} \frac{1}{N} \sum_{i=1}^{N} \underbrace{MSE(Y_i, h(\Phi(X_i), T_i))}_{\text{Outcome Loss}} + \lambda \underbrace{IPM(\Phi(X|T=1), \Phi(X|T=0))}_{\text{Dist. b/w T \& C covar. distributions}} + \alpha \underbrace{\mathcal{R}(h)}_{L_2} \tag{14}$$

where $\mathcal{R}(h)$ is a model complexity term and λ and α are hyperparameters.

overlap may be substantially biased in unknown ways.

[13]This kernel trick is also what makes support vector machines computationally tractable.
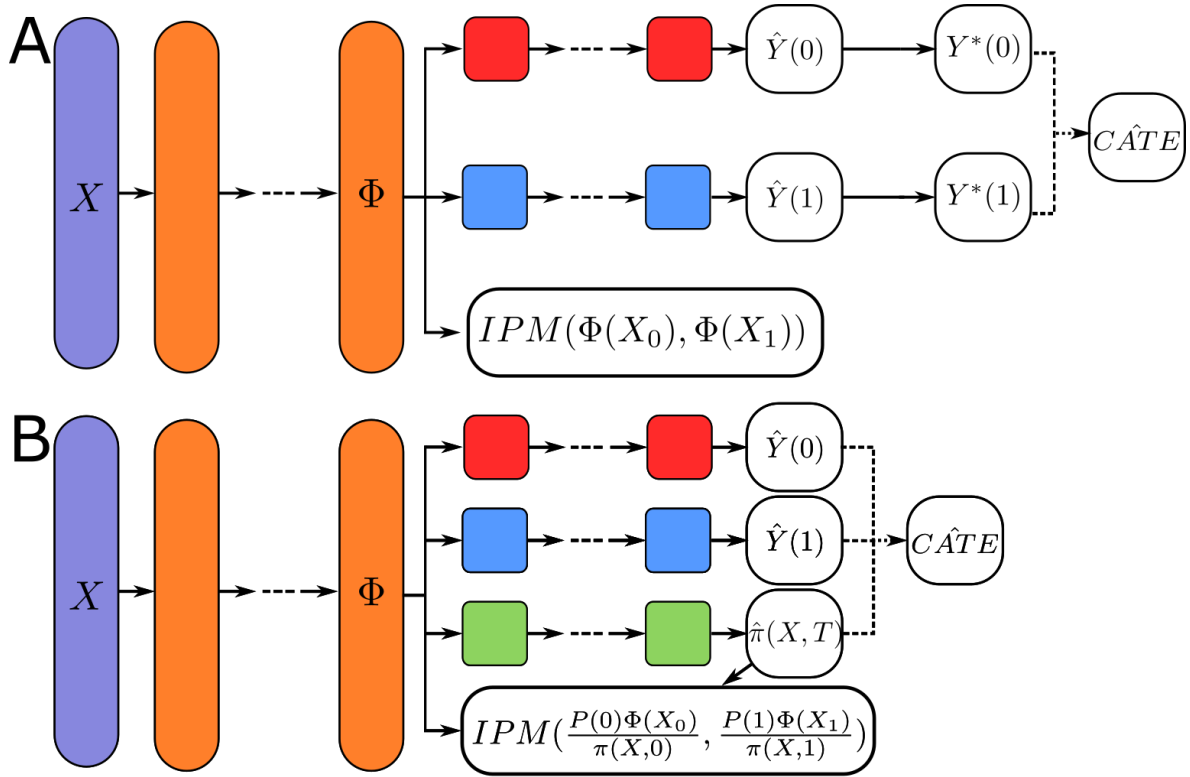
Figure 6: **A. CFRNet** architecture originally introduced in Shalit et al. (2017). CFRNet adds an additional integral probability metric (IPM) loss to TARNet to explicitly force representations of the treated and control covariates closer in representation space.
B. Weighted CFRNet adds a propensity score head to CFRNet to predict IPW-weighted outcomes. During training, the propensity score is used to reweight both the predicted outcomes $\hat{Y}(0)$ and $\hat{Y}(1)$, as well as the represented covariate distributions in calculation of the IPM loss. This allows the authors to provide consistency guarantees and relax the overlap assumption. Figures adapted from Johansson et al. (2020).

These two papers also make important theoretical contributions by providing bounds on the generalization error for the PEHE (Hill, 2011). In Shalit et al. (2017), they show that the PEHE is bounded by the sum of the factual loss, counterfactual loss, and the variance of the conditional outcome.

In Johansson et al. (2020), the authors introduce estimated IPW weights $\pi(\Phi(X), T)$ to CFRNet that are used within the IPM calculation to provide consistency guarantees (Fig. 6B). Theoretically, they also use these weights to relax the overlap assumption as long as the weights themselves obey the positivity assumption. From a practical standpoint, adding weights that are optimized smoothly across the whole dataset each epoch reduces noise created by calculating the IPM score in small batches. Weighted CFRNet minimizes the following loss function:

$$
\underset{h,\Phi,IPM,\pi,\lambda_h,\lambda_w}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} \underbrace{\frac{\hat{P}(T_i)}{\pi(\Phi(X_i), T_i)}}_{IPW} \cdot \underbrace{MSE(Y_i, h(\Phi(X_i), T_i))}_{\text{Outcome Loss}} + \lambda_h \underbrace{\mathcal{R}(h)}_{L_2 \text{ Outcome}} +
$$

$$
\alpha \cdot IPM(\underbrace{\underbrace{\frac{\hat{P}(1)}{\pi(\Phi(X,1))}}_{IPW} \cdot \Phi(X|T=1), \underbrace{\frac{\hat{P}(0)}{\pi(\Phi(X,0))}}_{IPW} \cdot \Phi(X|T=0))}_{\text{Distance between } IPW \text{ weighted T \& C covar. distributions}} + \lambda_\pi \underbrace{\frac{||\pi||_2}{N}}_{L_2 Var(\pi)} \tag{15}
$$

where $\mathcal{R}(h)$ is a model complexity term and λ_h, λ_π and α are hyperparameters. The final term is a regularization term on the variance of the weight parameters.

A.2.1 Extending Representation Balancing with Matching

Beyond IPMs, other approaches have directly embraced matching as a balancing strategy. Yao et al. (2018) train their TARNet on six point mini-batches of propensity score-matched units with additional reconstruction losses designed to preserve the relative distances between these points when projecting them into representation space. Schwab et al. (2018) takes an even simpler approach by feeding random batches of propensity-matched units to the TarNet outcome structure.

61

Appendix B   Model Selection Using the PEHE

In order to select hyperparameters in real data, Johansson et al. (2020) propose to use a matching variant of $PEHE$ with the nearest Euclidean neighbor of each unit i from the other treatment assignment group y_i^{nn} as a counterfactual. If we identify the nearest neighbor j of each unit i in representation space such that $t_j \neq t_i$ as

$$y_i^{nn}(1 - t_i) = \min_{j \in (1-T)} ||\Phi(x_i|t_i) - \Phi(x_j|1 - t_i)||_2$$

then,

$$PEHE_{nn} = \frac{1}{N} \sum_{i=1}^{N} (\underbrace{(1 - 2t_i)(y_i(t_i) - y_i^{nn}(1 - t_i))}_{CATE_{nn}} - \underbrace{(h(\Phi(x), 1) - h(\Phi(x), 0)))}_{\hat{CATE}}^2$$

If we take the square root of the $PEHE_{nn}$ then we get an approximation of the unit-level error.

The intuition behind $\sqrt{PEHE_{nn}}$ is solid. If our representation function Φ is truly learning to balance the treated and control distributions, $CATE_{nn}$ should coarsely measure it.

Appendix C   Recurrent Neural Networks (RNN)

Recurrent neural networks are a specialized architecture created for learning outcomes from sequential data (e.g. time series, biological sequences, text) (Fig. 7). In a classic RNN, each "unit" u in the network takes as input its own covariates X (or possibly a representation) and a representation produced by the previous unit, encoding cumulative information about earlier states in the sequence. These units are not just simple hidden layers: there is a set of weights within each unit for its raw inputs, the representation from the previous time step, and its outputs. Different RNN variants have different operations for integrating past representations with present inputs. Recurrent neural networks may be directed acyclic graphs or feedback on themselves. Commonly used variants include Gated Recurrent Unit networks (GRU) and Long-term Short-term memory networks (LSTM) (Cho et al., 2014; Hochreiter and Schmidhuber, 1997).
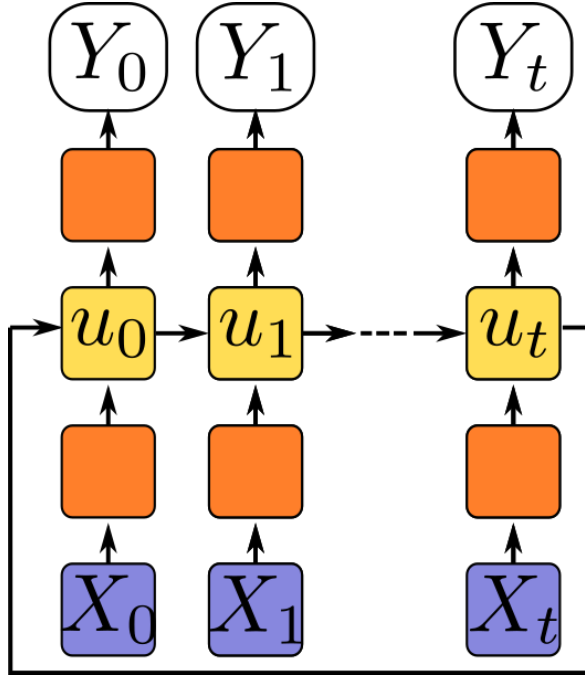
Figure 7: Recurrent neural network.

Appendix D   Generative Modeling through Adversarial Training

Adversarial training approaches include a wide variety of architectures where two networks or loss functions compete against each other. Adversarial approaches are inspired by Generative Adversarial Networks (GANs) (Box 9) (Goodfellow et al., 2014). In the machine learning literature on causal inference, adversarial training has been applied both to trade off outcome modeling and treatment modeling tasks during representation learning, as well as to trade off estimation and regularization of IPW weights. GANs have also been used directly as generative models for counterfactual and treatment effect distributions.

Box 9: Generative Adversarial Networks (GAN)

In GANs, two networks, a discriminator network D and a generator network G, play a zero-sum game like cops and robbers. The generator network's job is to learn a distribution from which the training data X could have credibly been generated. In each training batch, the generator produces a new outcome (originally images, but could be IPW weights, counterfactuals or treatment effects) by drawing a random noise sample from a known distribution Z (e.g. Gaussian) and transforming it into outcomes with the function $G(Z) = \hat{X}$. The discriminator's job is to learn a function $D(X) = P(X \ is \ real)$ that can distinguish whether the outcome is from the training data X, or whether it is a "fake" $\hat{X}$ created by the generator. The generator then receives a negative version of the discriminator's loss, a penalty that is proportional to how well it was able to "deceive" the discriminator. The discriminator's loss can be the log loss, Jensen-Shannon divergence (Goodfellow et al., 2014), the Wasserstein distance (Arjovsky et al., 2017; Gulrajani et al., 2017), or any number of divergences and IPMs. Formally, the generator attempts to minimize the following loss function,

$$\arg\min_{G} = \underbrace{\mathbb{E}_X}_{\text{real dist.}} [\underbrace{\mathcal{L}(D(X)}_{P(X \ is \ real)}] + \underbrace{\mathbb{E}_Z}_{\text{fake dist.}} [1 - \underbrace{\mathcal{L}(D(G(Z))}_{P(\hat{X} \text{is real})}]$$

where the first quantity is the discriminator's estimated probability data from X is indeed real, and the second quantity is the discriminator's estimate that a generated quantity from the distribution Z is real.

Because the discriminator is trying to catch the generator, its objective is to maximize the same function,

$$\arg\max_{D} = \underbrace{\mathbb{E}_X}_{\text{real dist.}} [\underbrace{\mathcal{L}(D(X)}_{P(X \text{is real})}] + \underbrace{\mathbb{E}_Z}_{\text{fake dist.}} [1 - \underbrace{\mathcal{L}(D(G(Z))}_{P(\hat{X} \text{is real})}]$$

In practice, the discriminator and the generator are trained either alternatingly or simultaneously, with the discriminator increasing its ability to discern between real and fake outcomes over time, and the generator increasing its ability to deceive the discriminator. The idea is that the adaptive loss function created by the discriminator can coax the generator out of local minima to generate superior outcomes. Results by these models have been impressive, and many of the fake portraits and "deepfake" videos circulating online in recent years are generated by this architectures. The advantage of GANs is that they can impressively learn very complex generative distributions with limited modeling assumptions. The disadvantage of GANs is that they are difficult and unreliable to train, often plateauing in local optima.

D.1 GANs as Generative Models of Treatment Effect Distributions (GAN-ITE)

Deep Dive: GANITE (Yoon et al. (2018)) Although a generative model of the treatment effect distribution is generally unknown, a natural application of GANs is to try to machine learn this distribution from data. GANITE uses two GANs: GAN_1, consisting of generator G_1 and discriminator D_1, to model the counterfactual distribution and GAN_2, consisting of generator G_2 and discriminator D_2, to model the $CATE$ distribution (Yoon et al., 2018) (Fig. 8). The training procedure for GAN_1 is as follows:

1. Taking X,T, and generative noise Z as input, generator G_1 generates both potential outcomes $\{\tilde{Y}(T), \tilde{Y}(1-T)\}$. A factual loss $MSE(Y(T), \tilde{Y}(T))$ is applied.

2. Create a new vector $\mathbf{C} = \{Y(T), \tilde{Y}(1-T)\}$ by combining the observed potential outcome and the counterfactual predicted by G_1.

3. Taking X and $\mathbf{C}$ as inputs, the discriminator D_1 rates each value in $\mathbf{C}$ for the probability that it is the observed outcome using the categorical cross entropy loss:

$$\mathcal{L}(D_1) = CCE(\{ \underbrace{P(\mathbf{C}_0 = Y(T))}_{\text{Prob first idx is real}}, \underbrace{P(\mathbf{C}_1 = Y(T))}_{\text{Prob sec idx is real}} \}, \{ \underbrace{\mathbf{C}_0 == Y(T)}_{\text{1 if idx 0 is real}}, \underbrace{\mathbf{C}_1 == Y(T)}_{\text{1 if idx 1 is real}} \}) \qquad (16)$$

4. This loss is then fed back to G_1 such that the total loss for the generator is now

$$\underset{G_1}{\arg\min} = MSE(Y(T), \tilde{Y}(T)) - \lambda \mathcal{L}(D_1) \qquad (17)$$

After generator G_1 is trained to completion, the authors use $\mathbf{C}$ as a "complete dataset" containing both a factual outcome and a counterfactual outcome to train GAN_2, which generates treatment effects:

1. Taking only X and generative noise Z as input, G_2 generates a new potential outcome vector $\mathbf{R} = \{\hat{Y}(T), \hat{Y}(1-T)\}$. G_2 receives an MSE loss to minimize the difference between its predictions and the "complete dataset" $\mathbf{C}$: $MSE(\mathbf{C}, \mathbf{R})$.

2. Discriminator D_2 takes X, $\mathbf{C}$, and $\mathbf{R}$ as inputs and estimates a probability that $\mathbf{C}$ is the "complete" dataset, and that $\mathbf{R}$ is the "complete dataset":

$$\mathcal{L}(D_2) = CCE(\{ \underbrace{P(\mathbf{C} = \mathbf{C})}_{\text{Prob } \mathbf{C} \text{ is "CD"}}, \underbrace{P(\mathbf{R} = \mathbf{C})}_{\text{Prob } \mathbf{R} \text{ is "CD"}} \}, \{ \underbrace{\mathbf{C} == \mathbf{C}}_{\text{1 if idx 0 is } \mathbf{C}}, \underbrace{\mathbf{C}_1 == Y(T)}_{\text{1 if idx 1 is } \mathbf{C}} \}) \qquad (18)$$
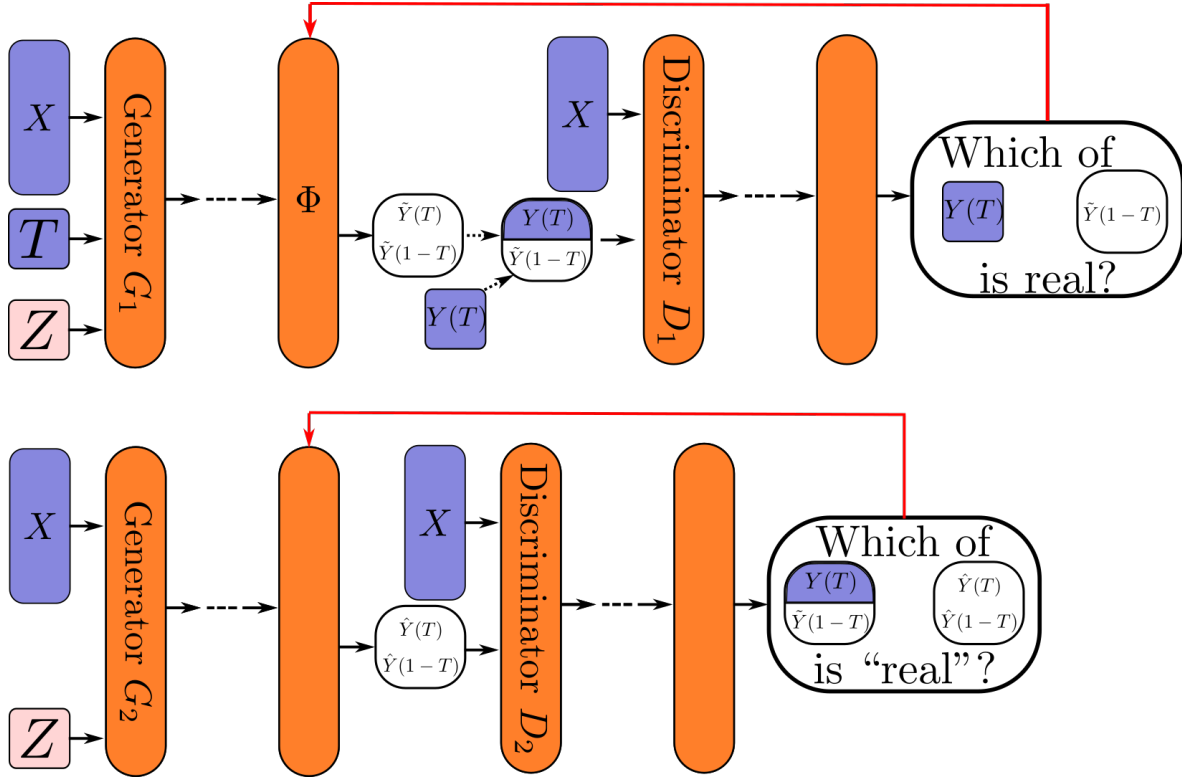
Figure 8: **GANITE** has two GANs. The first generator G_1 generates counterfactuals $\tilde{Y}(T)$. The discriminator D_1 attempts to discriminate between these predictions and real data $(Y(T))$. The second generator proposes pairs of potential outcomes $\hat{Y}(0)$ and $\hat{Y}(1)$ (i.e., treatment effects), a vector we call R. Discriminator D_2 attempts to discern between R and a "complete dataset" C created by pairing each observed/factual outcome $Y(T)$ with a synthetic outcome $\tilde{Y}(1-T)$ proposed by G_1. Although we do not show gradients in other figures, we make an exception for GANs (red line).

3. This loss is then fed back to the generator G_2 such that the total loss for the generator is now

$$\underset{G_2}{\arg\min} = MSE(\mathbf{C}, \mathbf{R}) - \lambda \mathcal{L}(D_2) \tag{19}$$

At the end of training, G_2 should be able to predict treatment effects with only covariates X and noise Z as inputs. An evolution of GANITE, SCIGAN, extends this framework to settings with more than one treatment and continuous dosages (Bica et al., 2020b).

D.2   Adversarial Representation Balancing

The use of the IPM loss in CFRNet (Shalit et al., 2017) may also be viewed as an adversarial approach in that the representation layers are forced to maximize performance on two competing tasks: predicting outcomes and minimizing an IPM. Rather than using an IPM loss, other authors have trained propensity score estimators that send positive (rather than negative) gradients back to the representation layers (Atan et al., 2018; Du et al., 2021).

Bica et al. (2020a) extend this approach to settings with treatment over time using a recurrent neural network. In their medical setting, decorrelating treatment from patient covariates and history allows them to estimate treatment effects at each individual snapshot.