Generative Modelling for Supervised, Unsupervised and Private Learning



James Jordon

Wolfson College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2021

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Mihaela van der Schaar who has been a pillar of opinion and guidance throughout my DPhil. Her support and grounding has been very important in both broadening my research interests and focusing my efforts. I would also like to thank everyone that has been part of the ML-AIM group during my time in it. In particular I would like to think Jinsung Yoon for his help in realising my ideas and always being a sounding board. I would like to thank Ioana Bica for our hard-worked collaborations and her friendship throughout. I would like to thank Bryan Lim, for countless discussions and his support early in my DPhil which helped me push through the tumultuous early months. I would also like to thank Ahmed Alaa, Alexis Bellot, Yao Zhang and Jeroen Berrevoets for countless discussions and paper reviews.

I would also like to thank Victor Prisacariu and Nicholas Hawes, who gave me feedback during my transfer and confirmation of status, and whose advice was important to hear. I would also like to thank Pawan Mudigonda for his support in helping me to continue my DPhil at Oxford.

Finally, I would like to thank my parents, Dawn and Tony, and my brother, Matty, for their support not just through this DPhil but throughout my life, which put me in a position to consider a DPhil in the first place.

Abstract

In this thesis we develop several state-of-the-art generative modelling-based approaches for a variety of supervised, unsupervised and private learning problems. In the (almost) supervised domain, we tackle the problems of treatment effect estimation, imputation and feature selection. For treatment effect estimation we begin by developing a GAN-based approach that generates the "missing" counterfactuals, which enables learning a fully supervised model. In SCIGAN, we then go on to adapt this method to the continuous-intervention setting, introducing novel generator and discriminator architectures to handle the continuous nature of the treatments. For imputation, we introduce GAIN, a GAN-based imputation approach that maximally leverages both the adversarial and generative nature of GANs to learn to impute missing data according to the ground truth distribution while simultaneously allowing for multiple imputation to improve reliability and robustness of the imputed data. Finally, for feature selection, we introduce KnockoffGAN, a GAN-based knockoff generation procedure which again leverages the adversarial nature of GANs to learn the appropriate "knockoff" distribution. In using a GAN, we are able to learn to generate knockoff non-parametrically and thus reliable performance on non-Gaussian data.

In the domain of private learning we address the problem of private synthetic data generation. This challenge has three key components: (1) generating faithful synthetic data; (2) ensuring privacy; and (3) measuring the quality of the generated data. In PATEGAN, we tackle all 3 of these problems by proposing a GAN-based approach to generating *differentially private* synthetic data. In addition, we introduce a new synthetic data metric, the Synthetic Ranking Agreement, which measures how well machine learning research on the synthetic data represents the same research on the real data. In the final paper presented, we present a new differential privacy building block - a differentially private classification/prediction algorithm, that builds upon the subsample-and-aggregate paradigm. The new method affords better privacy and utility than subsample-and-aggregate and can be used in place of PATE in the PATEGAN framework.

Contents

1 Introduction **1**
1.1 Generative Modelling 1
1.2 Contributions & Outline of Report 11
1.3 Publications 13

2 Treatment Effect Estimation **15**
2.1 GANITE: Estimation of Individualised Treatment Effects using Generative Adversarial Nets 16
2.2 Estimating the Effects of Continuous-valued Interventions using Generative Adversarial Networks 39

3 Imputation **69**
3.1 GAIN: Missing Data Imputation using Generative Adversarial Nets . 70

4 Feature Selection **89**
4.1 KnockoffGAN: Generating Knockoffs for Feature Selection using Generative Adversarial Networks 90

5 Private Learning **115**
5.1 PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees 116
5.2 Differentially Private Bagging: Improved utility and cheaper privacy than subsample-and-aggregate 138

6 Conclusion **154**
6.1 Possible Extensions 155

Bibliography **157**

Chapter 1

Introduction

Machine learning has proven to be a very powerful tool in a variety of domains. From learning to play Atari games [50] to identifying malignant skin lesions from photographs [23]. Much of the success can be attributed to the fact that deep neural networks are highly expressive, highly flexible functional approximators [1], capable of capturing complex non-linear dependencies within data. Generative Adversarial Networks [24] leverage this beautifully by not only using a network to perform the task in question (implicit generative modelling) but by defining the loss of the generative networks in terms of another network that is used to critique the generative network.

In this thesis we expand on this adversarial training in GANs. We leverage the fact that the discriminator is a classifier. In Chapters 2-4 we modify the classification objective from needing to classify real vs. generated samples to needing to classify whether individual dimensions of the input vector are real or generated. This allows us to learn to estimate individualised treatment effects (Chapter 2), impute missing data (Chapter 3) and generate knockoffs [5] for feature selection (Chapter 4). In Chapter 5 we shift the focus from investigating alternative distributions that can be learned with a GAN to generating private synthetic data, though once again we leverage the fact that the discriminator is a classifier to enforce differential privacy [20].

1.1 Generative Modelling

Data generating mechanisms can be modelled as (repeated) sampling from a distribution

$$\mathbf{x} \sim p(\mathbf{x})\,. \tag{1.1}$$

Generative modelling aims to model either the distribution

$$\hat{p}(\mathbf{x}) \approx p(\mathbf{x}) \tag{1.2}$$

or the sampling process itself

$$\hat{\mathbf{x}} \overset{\text{approx.}}{\sim} p(\mathbf{x})\,. \tag{1.3}$$

We refer to these two approaches as *explicit* and *implicit*, respectively. In the following discussion, let $\mathcal{D} = \{\mathbf{x}_i : i = 1, ..., N\}$ be a dataset consisting of N i.i.d. samples from $p(\mathbf{x})$.

1.1.1 Explicit models

The goal of an explicit model is to try and find an approximate density function $\hat{p} \approx p$. Approaches for doing so can be further broken down into 4 categories: (i) *parametric*; (ii) *semi-parametric*; (iii) *non-parametric*; and (iv) *approximate*.

Parametric density estimation Parametric models make several assumptions about the shape of the data in order to use a well-specified distribution and learn its parameters, typically through Maximum Likelihood Estimation (MLE) [51]. This involves maximising the log-likelihood of the data over the parameters of the distribution:

$$\theta^* = \arg\max_{\theta} \sum_{\mathbf{x} \in \mathcal{D}} \log p_\theta(\mathbf{x}) \tag{1.4}$$

for some appropriate parametric family of densities, $\{p_\theta : \theta \in \Theta\}$. We then obtain $\hat{p} = p_{\theta^*}$. Such an approach is highly restrictive and requires making several assumptions about the data a priori. This is at odds with the machine learning paradigm of trying to make as few assumptions about the data as possible.

Semi-parametric density estimation One step away from parametric estimators are semi-parametric estimators such as [56], or models that combine a mixture of simpler parametric models. Such methods exchange the ease of learning using MLE for improved flexibility, allowing a broader class of distributions to be learned using the same estimator. Though this comes at the cost of potentially expensive optimisation algorithms such as iterative Expectation-Maximization [18].

Non-parametric density estimation Offering the most flexibility among "exact" density estimation algorithms is those that employ non-parametric approaches. In doing so, very few assumptions are made about the structure of the underlying data, allowing complex real-world datasets to be modelled more accurately than is possible with parametric models. Kernel Density Estimation (KDE) [63, 10] involves estimating a density as a sum of kernels over all points in the dataset:

$$\hat{p}(\mathbf{x}) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right) \tag{1.5}$$

where K is some specified kernel, h is a smoothing parameter and d is the dimensionality of the data. Computing such estimates requires choosing both K and h, though some work has been done to allow h to be adaptive such as in k-nearest neighbour density estimation [44]. Kernel density estimators suffer from the curse of dimensionality [6]. Moreover, they are highly inappropriate when the data lies on a low-dimensional manifold of the feature space as they assign mass to all dimensions around the data points. To understand this, we note that sampling from the resultant KDE derived density involves a simple 2-step procedure: (1) sample a point from the dataset; (2) sample noise from the kernel and add this to the point from the dataset. With a Gaussian kernel, for example, this process amounts to adding Gaussian noise to a random sample from the data. Naturally, this is not guaranteed to produce a sample that actually lies in the support of the data distribution (consider images, for example).

Remark - Sampling from densities It should be noted that, for the applications we consider, our primary interest is in accessing samples from the learned distributions. Sampling from learned density functions can be done in several ways, including, but not limited to: *inversion sampling*, *rejection sampling*, *Markov-Chain-Monte-Carlo* [9] and, more recently there have been machine learning driven approaches such as [29]. If the sampling is not exact, however, then the produced samples will suffer from 2 sources of inaccuracy

$$\hat{\mathbf{x}} \overset{\text{approx.}}{\sim} \hat{p}(\mathbf{x}) \approx p(\mathbf{x})\,. \tag{1.6}$$

Moreover, a density contains more information than a set of samples does. Learning a density rather than learning to generate samples directly is thus both a harder problem and a more indirect solution to the problem we are most concerned with - generating additional samples.

Approximate density estimators Variational Autoencoders (VAEs) [40, 39] are a powerful generative framework. Success has been demonstrated in a wealth of applications from image generation [40, 30, 11] to causal inference [59] to imputation [46] and many other domains [61, 32, 65].

VAEs approach the problem of generative modelling by learning a two-part model comprising of an encoder and a decoder. The encoder maps the data into a latent domain, over which a prior distribution (typically Gaussian) is enforced, with the decoder then mapping latent samples to the original domain (e.g. pixel space for images). VAEs are trained by maximising the evidence lower bound (ELBO)

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|x)}[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \tag{1.7}$$

where p_θ denotes the (joint) distribution induced by the decoder (the generative model) and q_ϕ denotes the conditional distribution induced by the encoder (the variational model). The ELBO is a lower bound on the data likelihood, i.e. $\log p_\theta(\mathbf{x}) \geq \mathcal{L}_{\theta,\phi}(\mathbf{x})$, as such, it is hoped that by maximising this, some sort of approximate MLE is being performed. VAEs are easily sampled from by sampling from the prior distribution over the latent space, passing the latent sample through the decoder and sampling from the resultant distribution which is typically Gaussian [40].

Although VAEs admit exact estimates of the joint density $p(\mathbf{z},\mathbf{x})$, marginalisation over the latent variable, $\mathbf{z}$, is not trivial and often results in exact estimates of $p(\mathbf{x})$ being intractable. A common approach to obtaining density estimates from a VAE involves annealed importance sampling [66], resulting in approximate estimates of the VAE-induced density.

Research in VAEs is somewhat orthogonal to this thesis. Research has demonstrated that VAEs can be enhanced by the adversarial framework leveraged by GANs [47]. Moreover, [47] opens the door to adapting the research presented in this thesis to be used with a powerful VAE-GAN combo method. Like much research for GANs, methods for improving the underlying generative model are typically orthogonal to this thesis - any improvements can be incorporated provided they admit a learning paradigm that involves an adversarially trained classifier for learning the distribution. For this reason, [47] can be thought of as an extension of GANs as much as it can for VAEs. Importantly, the use of GANs throughout this thesis is motivated by the adversarial discriminator and the ability to modify it to adapt the GAN framework to a variety of problems. The two-player game in [47] closely resembles the adversarial game played by GANs and may admit similar modifications to be made appropriate for the problems presented.

1.1.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [24] are the building block for all but one paper presented within this thesis. Originally proposed in [24], they have since been heavily studied with many variations and improvements of the original GAN framework proposed. The contributions to the GAN literature can broadly be broken down into 3 categories: (a) *alternate loss functions*; (b) *improved training*; and (c) *applications of GANs*. The contributions of this thesis fall entirely into the latter category, though in Chapters 2-4 we introduce modifications of the standard GAN loss function that are problem-specific.

The original GAN framework as proposed in [24] involves training two networks, a generator, G, and a discriminator, D, in an adversarial fashion. By doing so, the generator, under the right conditions, will converge to generating samples from the true data distribution and as such can be used to sample from the true distribution. Importantly, GANs are an implicit model, for which the likelihood is not accessible. Though this makes analysis difficult, they allow for very simple sampling (which suffices in many use cases) and as has been demonstrated in the literature, GANs are capable of producing high-quality samples from complex data distributions. The original GAN loss involves training the GAN according to the following minimax game:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{1.8}$$

where $p(\mathbf{x})$ is the true data distribution and $p(\mathbf{z})$ is some prior noise distribution, typically uniform or Gaussian, that is fed into the generator to produce the generated samples $\hat{\mathbf{x}} = G(\mathbf{z})$. It can be shown that, at optimality, this minimax game is equivalent to minimising a Jensen-Shannon Divergence (JSD) between the real and generated data distributions [24]. Though the original paper considers only unconditional generation, the framework can be easily extended to learning a conditional distribution $p(\mathbf{x}|\mathbf{w})$ by feeding the conditioning variable, $\mathbf{w}$, into both the generator $G(\cdot, \mathbf{w})$ and discriminator $D(\cdot, \mathbf{w})$ [49]. A very successful alternative approach, AC-GAN [55], involves passing the conditioning variable to the generator, but not the discriminator and then asking the discriminator to predict the conditioning variable from the generated sample. The discussions that follow can all broadly be extended to the conditional case by adding the conditional variable as inputs to the involved networks.

1.1.2.1 Alternate Loss Functions

As noted above, the original GAN framework involves approximately optimising the Jensen-Shannon Divergence. A wealth of literature has been dedicated to determining methods for optimising other conventional metrics and divergences. The choice of metric can significantly affect the training and convergence of GANs [4, 72, 53, 8]. The JSD, for example, is constant when the two distributions in question have no overlap and thus has 0 gradient with respect to the generator parameters if the initialisation of the GAN network does not produce samples that lie within the support of the real data distribution [3]. This occurs very frequently when the true data manifold is a low-dimensional subspace of the full feature space, where by full feature space we mean the cross-product of the marginal feature spaces induced by projecting a data distribution down to each of the individual dimensions. This is easiest to visualise with images where the space of realistic images (not necessarily even limited to containing something specific) is a much smaller space than the space of all possible images that can be created by considering all possible pixel combinations. Below we discuss some of the more prominent loss-function investigations in the GAN literature.

Wasserstein GAN In Wasserstein GAN (WGAN) [4], they consider the Wasserstein distance, or Earth-Movers distance, given by:

$$W(p, q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||] \tag{1.9}$$

where $\Pi(p, q)$ is the set of all joint probability distributions $\gamma(x, y)$ that have marginal distributions p and q respectively [4]. Intuitively, this metric measures the amount of mass (probability mass) that needs to be moved (and how "far" it needs to be moved) to get from p to q. By leveraging the Kantorovich-Rubinstein duality [64], they rewrite the Wasserstein distance as

$$W(p, q) = \sup_{||f||\leq 1} \mathbb{E}_{\mathbf{x}\sim p}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim q}[f(\mathbf{x})] \tag{1.10}$$

where the supremum is over all 1-Lipschitz functions. By taking the supremum over all K-Lipschitz functions, one instead obtains $K \cdot W(p, q)$ and for the purpose of the optimisation, it therefore only matters that the supremum is taken over K-Lipschitz functions for some (potentially unknown) K. In the original WGAN paper they achieve this via weight-clipping. WGAN-GP [25] proposes adding a gradient-penalty (GP) term to the loss that encourages (but does not necessarily guarantee) that the critic is Lipschitz. This gradient penalty term leads to significant stability improvements and is widely used whenever a Wasserstein GAN is implemented [25, 35, 36, 15].

f-GAN f-GAN [53] generalises the original GAN framework by replacing the approximate Jensen-Shannon Divergence loss with the more general f-divergence. Given distributions p and q, the f-divergence is given by

$$D_f(p||q) = \mathbb{E}_{x\sim q}\left[f\left(\frac{p(x)}{q(x)}\right)\right] \tag{1.11}$$

where $f : \mathbb{R}^+ \to \mathbb{R}$ is some convex, lower-semicontinuous function with $f(1) = 0$. The JSD-approximate GAN loss from [24] is obtained as a special case when $f(u) = u\log u - (u+1)\log(u+1)$. Similar in spirit to WGAN, [53] leverages a duality via Fenchel conjugates [27], f^*, that allow optimising over the variational lower bound on the f-divergence [52] given by

$$D_f(p||q) \geq \sup_{T\in\mathcal{T}} \mathbb{E}_{x\sim p}[T(x)] - \mathbb{E}_{x\sim q}[f^*(T(x))] \tag{1.12}$$

where $\mathcal{T}$ is an arbitrary class of functions [53]. Through this lens of f-divergences, they investigate several known divergences such as Kullback-Leibler (KL), Reverse KL and Pearson χ^2.

Maximum mean discrepancy GAN In the same spirit as f-GAN, [8] approaches the GAN framework with an attempt to understand how integral probability metrics (IPMs) play a role, with the Wasserstein distance being a specific instance of an IPM. In [8], they focus on an alternative IPM, the maximum mean discrepancy (MMD) which was first proposed for GAN training in [43] and [21]. An IPM is defined, via some class of witness functions $\mathcal{F}$, by

$$D_{\mathcal{F}}(p, q) = \sup_{f\in\mathcal{F}} \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{x\sim q}[f(x)]\,. \tag{1.13}$$

By setting $\mathcal{F}$ to be the class of all 1-Lipschitz functions, we recover the Wasserstein distance. For the maximum mean discrepancy, $\mathcal{F}$ is set to be the unit ball in a reproducing kernel Hilbert space [8]. In particular, they show that with smaller discriminator architectures (which thus result in faster training), they can achieve similar results to WGAN-GP.

Much of the work presented in this thesis lies in a somewhat orthogonal direction to the above work on GANs. Although in most papers we do introduce a modified GAN loss, it is for the purpose of addressing a specific problem (imputation, treatment effect estimation, knockoff generation) which can not be achieved with a vanilla GAN

architecture. More importantly, we directly modify the adversarial classifier that the vanilla GAN framework proposes. Such modifications are not possible with WGAN, f-GAN or MMDGAN. That is not to say that these frameworks are not extensible to the problems we consider, and that would indeed be an interesting direction for future research on the use of GANs in domains beyond image generation. If such techniques can be made to work in the settings we consider without sacrificing too much then the improvements that WGAN, for example, makes over vanilla GAN could provide significant improvements for our problems too.

1.1.2.2 Improved Training

Instability in GAN training is a well-known problem [60, 25, 62, 12]. There are 3 main forms of GAN instability: (1) vanishing gradients; (2) mode collapse; and (3) failure to converge.

Vanishing Gradients Vanishing gradients are primarily a problem in the original GAN framework [24]. As noted above, the JSD is constant for two distributions with no overlap and as such the discriminator produces no gradient information for the generator. The original paper proposes a modified loss for the generator given by

$$\mathcal{L}(\theta_G) = -\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(D(G(\mathbf{z})))] \tag{1.14}$$

which corresponds to trying to make the discriminator classify the generator samples as more real, as opposed to trying to make the discriminator classify them as less fake. They note that the original loss $\log(1 - D(G(\mathbf{z})))$ saturates early on in training when G is poor (because D can distinguish samples well), whereas $\log(D(G(\mathbf{z})))$ provides much stronger gradients early on. In fact, WGAN was proposed in part to address this problem - a WGAN discriminator can be trained to optimality without introducing vanishing gradients [4].

Mode Collapse Mode collapse [62, 60] occurs when the generator learns to generate a few non-diverse samples from some of the training data modes but misses other modes of the training data completely. This problem is prevalent in the standard GAN formulation [24]. Each generator update encourages the generator to find the single most plausible output. Ideally, the iterative training procedure means that the discriminator updates will cause the most plausible output to constantly be changing, but if the discriminator gets stuck in a local minima, then the generator can easily find the most plausible output. Using a WGAN reduces the problem - the critic

can be trained to optimality in between each generator input. Unrolled GANs [48] propose an unrolled optimisation scheme for the discriminator that involves updating the generator not only according to the current discriminator, but also according to how the discriminator will change in response to the update.

Failure to converge Convergence failure in GANs occurs when the discriminator and generator fail to find equilibrium. Standard gradient descent optimisers such as Adam [38] were not designed for the non-convex saddle optimisation required by GANs. [26] propose a "two time-scale update rule" that involves separate learning rates and regimes for the discriminator and generator, and provably demonstrate convergence. In [35] they propose a model that progressively grows during the course of training - the GAN starts by generating and discriminating low resolution images and then gradually increases the resolution by adding (de-)convolutional layers. This stabilises training and helps address mode collapse, with [35] demonstrating better variability in the produces samples than a standard WGAN. Note though, that the technique is specific to image data and does not generalise to tabular data.

Much of the work in stabilising GANs and their training can be used in much the same way to stabilise the training of the GANs we present in this thesis. We do not explicitly investigate these techniques as applied to our frameworks in this thesis, as the focus is on fundamentally adapting the GAN framework to new problem settings. We do not anticipate that such techniques would be difficult to incorporate but leave it as an area for potential future research and/or to the research engineers tasked with implementing such ideas in the real world.

1.1.2.3 Applications of GANs

GANs have been demonstrated to be successful in a wide variety of applications. Of course, the primary area of interest has been image generation; [24, 35, 36, 4] are just some of the papers already mentioned that focus almost entirely on image generation. However, the GAN approach is not only suitable for images. Of particular interest to us in this thesis is the usability of GANs in problems that present themselves in general medical data (i.e. tabular data). In [14] the authors demonstrate that a GAN is capable of learning to generate synthetic medical records and even perform a (limited) evaluation of the generated data from a privacy perspective. In [22] they extend the GAN framework to the time-series setting, once again with a focus on medical data.

Orthogonal to the *types* of data that GANs have been used for, is the range of problems that GANs have been applied to. [17, 54, 41, 42] investigate the idea of using GANs for semi-supervised learning. [15, 74, 2, 37] utilise GANs for unpaired image-to-image domain translation and cross-domain relation discovery. [13, 58, 19] focus on leveraging GANs for representation learning, with [13] in particular demonstrating the ability to learn *disentangled representations* which are useful in themselves but can also be used to control the generative procedure. [67, 68, 45] add privacy to the GAN training loop so that the learned generator can be used to generate private synthetic data. In this thesis we extend the set of explored problems to treatment effect estimation [70, 7], imputation [69] and feature selection [33], in addition to adding to the literature on private data generation in [34].

1.1.3 Conditional generative modelling

The discussion thus far has been grounded in the goal of generating samples from the full joint distribution $p(\mathbf{x})$. This is of particular interest to us in Chapter 5 in which we investigate the use of GANs for private synthetic data generation. For the other problems we consider, however, we are interested in conditional generative modelling, in which we hope to learn to generate samples

$$\hat{\mathbf{x}} \overset{\text{approx.}}{\sim} p(\mathbf{x}|\mathbf{w}) \tag{1.15}$$

where $\mathbf{w}$ is some conditioning variable. Many of the paradigms discussed above can be adapted to the conditional setting: conditional KDE [28], conditional VAE [73, 46], conditional GAN [49]. The adaptation for KDEs involves estimating a double-kernel

$$\hat{p}(\mathbf{x}|\mathbf{w}) = \frac{\sum_{i=1}^{N} K_{h_1}(\mathbf{x} - \mathbf{x}_i) K_{h_2}(\mathbf{w} - \mathbf{w}_i)}{\sum_{i=1}^{N} K_{h_2}(\mathbf{w} - \mathbf{w}_i)} \tag{1.16}$$

where h_1, h_2 are two different smoothing parameters that must satisfy some conditions [28] in order to ensure consistency of the estimator.

For both VAE and GAN models, conditioning on $\mathbf{w}$ involves passing $\mathbf{w}$ as an additional input to the networks concerned. For vanilla GANs, for example, we have the objective adjusted to

$$\min_G \max_D \mathbb{E}_{\mathbf{x},\mathbf{w}\sim p(\mathbf{x},\mathbf{w})}[\log D(\mathbf{x},\mathbf{w})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}),\mathbf{w}\sim p(\mathbf{w})}[\log(1 - D(G(\mathbf{z},\mathbf{w}),\mathbf{w}))]\,. \tag{1.17}$$

In the GAN literature this has primarily been used to generate images conditional on a specific class-label [49]. This conditioning has also been used, however, to generate

descriptions from images [16], for image-to-image translation [31, 15] and many other applications.

In Chapter 2 we consider the problem of generating potential outcomes for treatments from pre-treatment covariates. In Chapter 3 we consider the problem of imputation, which amounts to generating the missing data components from the observed data components. In Chapter 4, we generate knockoffs which are inherently paired with observations (and thus the knockoffs are conditionally generated based on observed data).

1.2 Contributions & Outline of Report

1.2.1 Research Directions

This thesis focuses on the development of state-of-the-art GAN-based models for 3 key problem areas - treatment effect estimation, imputation & feature selection (data pre-processing), and synthetic data generation.

Adapting GANs Our primary focus in this thesis is in adapting the GAN framework to a variety of problems. In Chapters 2-4 we modify the GAN discriminator such that it is no longer performing sample-wide binary classification but rather attempting to distinguish individual components of the sample as real or generated. This manifests itself in a variety of ways even across the 3 problems we consider: (i) treatment effect estimation; (ii) imputation; and (iii) feature selection. In the treatment effect estimation problem, we observe only 1 of the (potentially many) potential outcomes, and we task the discriminator with identifying this outcome from among generated counterfactuals. In the imputation problem, the discriminator observes an imputed data vector and is tasked with identifying which of the components are real and which were generated. Finally, in the feature selection problem, to generate realistic knockoffs the discriminator is given a randomly swapped feature-knockoff vector, and tasked with identifying which components have been swapped.

Private Synthetic Data Generation Of adjacent interest is the problem of generating private synthetic data. Once again, we leverage the fact that a vanilla GAN discriminator is a classifier to train a differentially-private GAN using the subsample-and-aggregate mechanism. Moreover, we investigate an improvement over the well-established subsample-and-aggregate mechanism which has implied improvements for our GAN framework.

1.2.2 Thesis Contributions & Outline

We split our work on Adapting GANs into 3 chapters (Chapters 2-4), one for each of the problems we consider. Chapter 5 is then dedicated to the problem of synthetic data generation and privacy. Finally in Chapter 6 we provide conclusions and possible directions for future research that would extend the work in this thesis.

Treatment Effect Estimation: In Chapter 2, we propose two new state-of-the-art deep learning architectures for treatment effect estimation in both the discrete and continuous settings. With GANITE [70], we introduce a GAN-based counterfactual estimator that transforms the problem of potential outcome estimation into a supervised problem by generating the counterfactual outcomes for each data point. In doing so, we remove the effects of assignment bias and allow for standard supervised methods to be used to obtain treatment effect outcomes. In SCIGAN [7], we develop this idea further, by extending to the setting in which treatments are assigned with a dosage parameter and there are potentially several such treatments. In doing so, we introduce a novel hierarchical discriminator architecture that is designed for the problem of continuous-intervention estimation. In SCIGAN we develop theory for both SCIGAN and GANITE, demonstrating that the objective is minimised by a desirable generator.

Imputation: In Chapter 3, we propose a state-of-the-art GAN-based architecture for imputation that is capable of multiple imputation through its generative nature. Moreover, we provide theoretical underpinnings for the methodology and derive necessity for our introduced *hint* vector, which speeds and stabilises convergence.

Feature Selection: In Chapter 4, we develop a state-of-the-art knockoff [5] generation method, which, to our knowledge, was the first capable of removing distributional assumptions on the data. We draw on machine learning works on mutual information estimation to enable maximal power in the knockoff procedure. Once again, we provide theoretical justification for our methodology, demonstrating that under standard GAN assumptions [24], our method produces valid knockoffs.

Privacy & Synthetic Data Generation: In Chapter 5, we begin with PATE-GAN, a novel approach for differentially private synthetic data generation. Once again leveraging the fact that the discriminator is a classifier, we train PATEGAN using the subsample-and-aggregate methodology proposed in [57]. In addition, we

introduce a new metric for measuring the quality of synthetic data, which we refer to as the synthetic ranking agreement. This metric concerns itself with ensuring that synthetic data reflects the real data for the purposes of machine learning research - if method A outperforms method B on the real data, then it also should on the synthetic data. In the second half of Chapter 5, we introduce a new differentially private classification methodology by modifying subsample-and-aggregate. We investigate the effects of partitioning the data multiple times rather than just once, and demonstrate that both utility and privacy increase as a result, though at the cost of higher computational cost.

1.2.2.1 Presentation Format

This thesis is presented in the *integrated thesis format*[1], with the main chapters consisting of papers that have been accepted for publication. To ensure compliance with any copyright restrictions, the papers have been reproduced in the original format used in their pre-print versions.

1.3 Publications

The work presented in this thesis has been submitted for publication to the venues below, with their respective statuses of acceptance detailed inline. * indicates joint authorship.

Ch. 2: Treatment Effect Estimation

- Jinsung Yoon, **James Jordon**, and Mihaela van der Schaar. GANITE: Estimation of Individualised Treatment Effects using Generative Adversarial Nets. In *International Conference on Learning Representations*, 2018. Weblink: `https://openreview.net/references/pdf?id=ryHhC0MDf`. [Accepted]

- Ioana Bica*, **James Jordon***, and Mihaela van der Schaar. Estimating the effects of continuous-valued interventions using generative adversarial networks. In *Advances in Neural Information Processing Systems*, 2020. Weblink: `https://arxiv.org/pdf/2002.12326.pdf`. [Accepted]

[1] `https://www.mpls.ox.ac.uk/graduate-school/information-for-postgraduate-research-students/submitting-your-thesis`

Ch. 3: Imputation

- Jinsung Yoon*, **James Jordon***, and Mihaela van der Schaar. GAIN: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning*, 2018. Weblink: `http://proceedings.mlr.press/v80/yoon18a/yoon18a.pdf`. [Accepted]

Ch. 4: Feature Selection

- **James Jordon**, Jinsung Yoon and Mihaela van der Schaar. KnockoffGAN: Generating Knockoffs for Feature Selection using Generative Adversarial Networks. In *International Conference on Learning Representations*, 2019. Weblink: `https://openreview.net/pdf?id=ByeZ5jC5YQ`. [Accepted]

Ch. 5: Private Learning

- **James Jordon***, Jinsung Yoon* and Mihaela van der Schaar. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *International Conference on Learning Representations*, 2019. Weblink: `https://openreview.net/pdf?id=S1zk9iRqF7`. [Accepted]

- **James Jordon**, Jinsung Yoon and Mihaela van der Schaar. Differentially Private Bagging: Improved utility and cheaper privacy than subsample-and-aggregate. In *Advances in Neural Information Processing Systems*, 2019. Weblink: `https://openreview.net/pdf?id=HkljOVBgUH`. [Accepted]

A subset of the work in PATEGAN was also presented at the following workshop:

- **James Jordon**, Jinsung Yoon and Mihaela van der Schaar. Measuring the quality of Synthetic data for use in competitions. In *Proceedings of KDD Workshop on Machine Learning for Medicine and Healthcare*, 2018. Weblink: `https://arxiv.org/pdf/1806.11345.pdf`. [Accepted]

Chapter 2

Treatment Effect Estimation

Publications Included

- Jinsung Yoon, **James Jordon**, and Mihaela van der Schaar. GANITE: Estimation of Individualised Treatment Effects using Generative Adversarial Nets. In *International Conference on Learning Representations*, 2018. Weblink: `https://openreview.net/references/pdf?id=ryHhC0MDf`. [Accepted]
- Ioana Bica*, **James Jordon***, and Mihaela van der Schaar. Estimating the Effects of Continuous-valued Interventions using Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, 2020. Weblink: `https://arxiv.org/pdf/2002.12326.pdf`. [Accepted]

GANITE: ESTIMATION OF INDIVIDUALIZED TREATMENT EFFECTS USING GENERATIVE ADVERSARIAL NETS

Jinsung Yoon
Department of Electrical and Computer Engineering
University of California, Los Angeles
Los Angeles, CA 90095, USA
jsyoon0823@g.ucla.edu

James Jordon
Department of Engineering Science
University of Oxford
Oxford, UK
james.jordon@wolfson.ox.ac.uk

Mihaela van der Schaar
Department of Engineering Science, University of Oxford, Oxford, UK
Alan Turing Institute, London, UK
mihaela.vanderschaar@eng.ox.ac.uk

ABSTRACT

Estimating individualized treatment effects (ITE) is a challenging task due to the need for an individual's potential outcomes to be learned from biased data and without having access to the counterfactuals. We propose a novel method for inferring ITE based on the Generative Adversarial Nets (GANs) framework. Our method, termed Generative Adversarial Nets for inference of Individualized Treatment Effects (GANITE), is motivated by the possibility that we can capture the uncertainty in the counterfactual distributions by attempting to learn them using a GAN. We generate proxies of the counterfactual outcomes using a counterfactual generator, $\mathbf{G}$, and then pass these proxies to an ITE generator, $\mathbf{I}$, in order to train it. By modeling both of these using the GAN framework, we are able to infer based on the factual data, while still accounting for the unseen counterfactuals. We test our method on three real-world datasets (with both binary and multiple treatments) and show that GANITE outperforms state-of-the-art methods.

1 INTRODUCTION

Individualized treatment effects (ITE) estimation using observational data is a fundamental problem that is applicable in a wide variety of domains. For instance, (1) in understanding the heterogeneous effects of drugs (Shalit et al. (2017); Alaa & van der Schaar (2017); Alaa et al. (2017)); (2) in evaluating the effect of a policy on unemployment rates (LaLonde (1986); Smith & Todd (2005)); (3) in verifying which factor causes a certain disease (Höfler (2005)) and (4) estimating the effects of pollution on the weather (Hannart et al. (2016)).

As explained in Spirtes (2009), the problem of ITE estimation differs from the standard supervised learning problem. First, among the potential outcomes, only the factual outcome is actually observed (revealed), counterfactual outcomes are not observed and so the entire vector of potential outcomes can never be obtained. Second, unlike randomized controlled trials (RCT), observational studies are prone to treatment selection bias. For instance, left ventricular assist device (LVAD) treatment is mostly applied to high-risk patients with severe cardiovascular diseases before heart transplantation, the distribution of features among these patients will be significantly different to the distribution among non-LVAD treated patients (Kirklin et al. (2010)). The sample distribution can vary drastically across different choices of treatments and therefore, if we were to apply a supervised learning framework for each treatment separately, the learned models would not generalize well to the entire population.

Classical works in this domain solved the problem of estimating the average treatment effects from observational data (Dehejia & Wahba (2002b); Lunceford & Davidian (2004)). These works account

for the selection bias using propensity scores (the estimated probability of receiving a treatment) to create unbiased estimators of the average treatment effect. Dehejia & Wahba (2002b) used a one-to-one matching methodology to pair treated and control patients with similar features while Lunceford & Davidian (2004) used propensity scoring weighing to account for the selection bias. More recent works focus on individualized treatment effects (Chipman et al. (2010); Wager & Athey (2017); Athey & Imbens (2016); Lu et al. (2017); Alaa & van der Schaar (2017); Porter et al. (2011); Johansson et al. (2016); Alaa et al. (2017); Louizos et al. (2017); Shalit et al. (2017)). Detailed qualitative comparisons to these works will be discussed in the next subsection and numerical comparisons can be found in Section 5.

In this paper, we propose a novel approach that attempts to not only fit a model to the observed factual data, but also account for the unseen counterfactual outcomes. We view the factual outcome as an observed label and consider the counterfactual outcomes to be missing labels. Missing labels are generated by the well-known Generative Adversarial Nets (GAN) framework (Goodfellow et al. (2014)). More specifically, the counterfactual generator of GANITE attempts to generate counterfactual outcomes in such a way that when given the combined vector of factual and generated counterfactual outcomes the discriminator of GANITE cannot determine which of the components is the factual outcome. With the complete labels (combined factual and estimated counterfactual outcomes), the ITE estimation function can then be trained for inferring the potential outcomes of the individual based on the feature information in a supervised way. By also modelling this ITE estimation function using a GAN framework, we are able not only to predict the expected outcomes but also provide confidence intervals for the predictions, which is very important in, for example, the medical setting.

Unlike many other state-of-the-art methods, our method naturally extends to - and in fact is defined in the first place for - any number of treatments. We conduct experiments with three real-world observational datasets (with both binary and multiple treatments), and GANITE outperforms state-of-the-art methods.

1.1 Related Works

Previous works on ITE estimation can be divided into three categories. In the first, a separate model is learned for each treatment; this approach does not account for selection bias and so each model learned will be biased toward the distribution of that treatment's population. In the second, the treatment is considered a feature, with one model learned for everything, and the mismatch between the entire sample distribution and treated and control distributions is adjusted in order to account for selection bias. For instance, Chipman et al. (2010); Wager & Athey (2017); Athey & Imbens (2016); Lu et al. (2017) used tree-based models, Porter et al. (2011) used doubly-robust methods, Dehejia & Wahba (2002b); Lunceford & Davidian (2004), k-nearest neighbor (kNN) Crump et al. (2008) used propensity and matching based methods, and Johansson et al. (2016); Shalit et al. (2017) used deep learning approaches to solve the ITE problem under this one model methodology. Inherent to the approach of learning a balanced representation is that the representation must trade off between containing predictive information and reducing biased information. This is because often it will be the case that information that is biased is also highly predictive (in fact in the medical setting this is precisely why it is biased - because the doctors will assign treatments based on predictive features). On the other hand, our framework is not forced to make this information trade-off - the dataset we learn our final ITE estimator on contains the original dataset, and so contains at least as much information as that one. In the experiment section, we show that our proposed framework outperforms Shalit et al. (2017), particularly when the bias is high. In the third category, Alaa & van der Schaar (2017); Alaa et al. (2017) used a multi-task model approach. Alaa et al. (2017) used multi-task neural nets to estimate (1) the selection bias, (2) the controlled outcome and (3) the treated outcome with shared layers across these three tasks. Alaa & van der Schaar (2017) used a Gaussian Process approach in the multi-task model setting. Our work is perhaps most similar to Alaa & van der Schaar (2017) since there, too, they attempted to account for the counterfactuals and were similarly able to provide confidence in their estimates using credible intervals. They were able to access counterfactuals through a posterior distribution which was then accounted for in the learning of their model.

2 PROBLEM FORMULATION: ESTIMATION OF INDIVIDUALIZED TREATMENT EFFECTS

Let $\mathcal{X}$ denote the s-dimensional feature space and $\mathcal{Y}$ the set of possible outcomes. Consider a joint distribution, μ, on $\mathcal{X} \times \{0,1\}^k \times \mathcal{Y}^k$ where k is the number of possible treatments. Suppose that $(\mathbf{X}, \mathbf{T}, \mathbf{Y}) \sim \mu$. We call $\mathbf{X} \in \mathcal{X}$ the (s-dimensional) feature vector, $\mathbf{T} \equiv (T_1, ..., T_k) \in \{0,1\}^k$ the treatment vector and $\mathbf{Y} \equiv (Y_1, ..., Y_T) \in \mathcal{Y}^k$ the vector of potential outcomes (or the *Individualized Treatment Effects (ITE)*). We assume that (with probability 1), there is precisely one non-zero component of $\mathbf{T}$ and we denote by η the index of this component. Denote by $\mu_{\mathbf{X}}$ the marginal distribution of $\mathbf{X}$ and by $\mu_{\mathbf{Y}}(\mathbf{x})$ the conditional distribution of $\mathbf{Y}$ given $\mathbf{X} = \mathbf{x}$, for $\mathbf{x} \in \mathcal{X}$ (marginalized over $\mathbf{T}$). This setting is known as the Rubin-Neyman causal model (Rubin (2005)).

We introduce two[1] assumptions about the distribution μ in the Rubin-Neyman causal model.

Assumption 1. ***(Overlap)*** *For all $\mathbf{x} \in \mathcal{X}$, for all $i \in \{1, ..., k\}$,*

$$0 < \mathbb{P}(T_i = 1 | \mathbf{X} = \mathbf{x}) < 1.$$

This assumption ensures that at every point in the feature space, there is a non-zero probability of being given treatment i for every i.

Assumption 2. ***(Unconfoundedness)*** *Conditional on $\mathbf{X}$, the potential outcomes, $\mathbf{Y}$, are independent of $\mathbf{T}$,*

$$\mathbf{Y} \perp\!\!\!\perp \mathbf{T} | \mathbf{X}.$$

This assumption is also referred to as *no unmeasured confounding* and requires that all *joint* influences on $\mathbf{Y}$ and $\mathbf{T}$ are measured. Note that this assumption means that $\mu_{\mathbf{Y}}(\mathbf{x})$ no longer needs to be marginalized over $\mathbf{T}$, since, under this assumption, they are independent.

Assume now that we observe samples of $(\mathbf{X}, \mathbf{T}, Y_\eta)$ (whose joint distribution we denote by μ_f), so that our dataset, $\mathcal{D}$, is given by $\mathcal{D} = (\mathbf{x}(n), \mathbf{t}(n), y_{\eta(n)}(n))_{n=1}^N$. Importantly, we only observe the component of the potential outcome vector that corresponds to the assigned treatment, we call this the *factual* outcome, and refer to unobserved potential outcomes as counterfactual outcomes or just *counterfactuals*. We denote by $y_f(n)$ and $\mathbf{y}_{cf}(n)$ the factual outcome and (vector of) counterfactual outcome(s), respectively. From this point forward, we omit the dependence on n for ease of notation.

In this setting, we wish to be able to draw samples from $\mu_{\mathbf{Y}}(\mathbf{x})$ for any $\mathbf{x} \in \mathcal{X}$. We measure the performance of the generator, $\mathbf{I}(\mathbf{x})$, using two different metrics depending on whether $k = 2$ (i.e. binary treatments) or $k > 2$ (i.e. multiple treatments).

For $k = 2$ we use the expected Precision in Estimation of Heterogeneous Effects, ϵ_{PEHE}, introduced in Hill (2011), given by:

$$\epsilon_{PEHE} = \mathbb{E}_{\mathbf{x} \sim \mu_{\mathbf{X}}} \left[\left(\mathbb{E}_{\mathbf{y} \sim \mu_{\mathbf{Y}}(\mathbf{x})}[y_1 - y_0] - \mathbb{E}_{\hat{\mathbf{y}} \sim \mathbf{I}(\mathbf{x})}[\hat{y}_1 - \hat{y}_0] \right)^2 \right]. \tag{1}$$

For $k > 2$ we use the expected mean squared error:

$$\epsilon_{MSE} = \mathbb{E}_{\mathbf{x} \sim \mu_{\mathbf{X}}} \left[|| \mathbb{E}_{\mathbf{y} \sim \mu_{\mathbf{Y}}(\mathbf{x})} \left[\mathbf{y} \right] - \mathbb{E}_{\hat{\mathbf{y}} \sim \mathbf{I}(\mathbf{x})} [\hat{\mathbf{y}}] ||_2^2 \right] \tag{2}$$

where $|| \cdot ||_2$ is the standard ℓ_2-norm in $\mathbb{R}^k$.

In order to achieve this goal, we separate the problem into two parts. First, we attempt to generate proxies for the unobserved counterfactual outcomes using a counterfactual generator ($\mathbf{G}$) to create a *complete* dataset. Then, using this proxy dataset, we learn the ITE generator, $\mathbf{I}$.

3 GANITE: GENERATIVE ADVERSARIAL NETS FOR INFERENCE OF INDIVIDUALIZED TREATMENT EFFECT ESTIMATION

3.1 OVERVIEW

The objective of GANITE is to generate potential outcomes for a given feature vector $\mathbf{x}$. However, due to the lack of counterfactual outcomes we are unable to learn the distribution of potential outcomes directly. To account for these counterfactuals, we first attempt to generate samples, $\tilde{\mathbf{y}}_{cf}$, using

[1] All works cited in the Related Works section make this assumption.

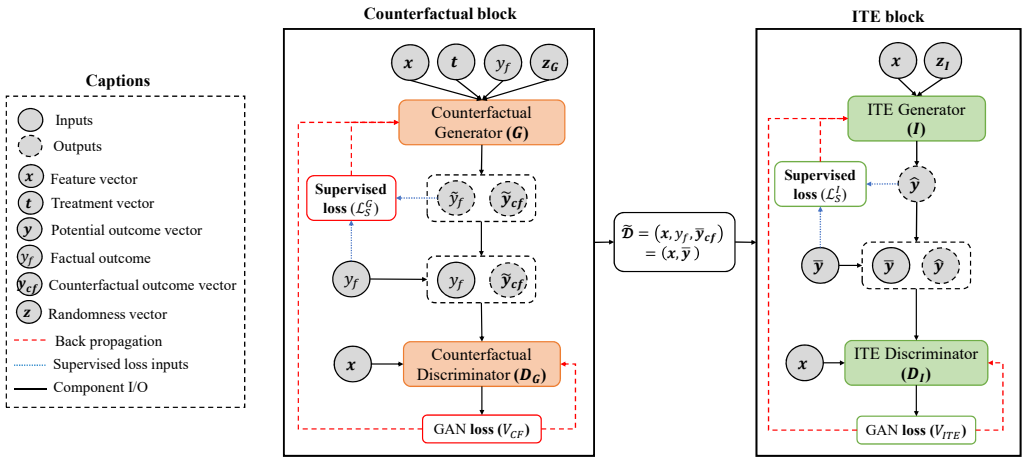


Figure 1: Block Diagram of GANITE ($\bar{\mathbf{y}}$ is sampled from $\mathbf{G}$ after $\mathbf{G}$ has been fully trained). $\mathbf{G}, \mathbf{D_G}, \mathbf{D_I}$ are only operating during training, whereas $\mathbf{I}$ operates both during training and at runtime.

a counterfactual generator, $\mathbf{G}$, from the distribution $\mu_{\mathbf{Y}_{cf}}(\mathbf{x}, \mathbf{t}, y_f)$ (the conditional distribution of the counterfactual outcomes, $\mathbf{Y}_{cf}$ given that $\mathbf{X} = \mathbf{x}$, $\mathbf{T} = \mathbf{t}$ and $Y_\eta = y_f$) for each sample in our dataset. We can then combine these *proxy counterfactuals* with the original dataset to obtain a *complete* dataset $\tilde{\mathcal{D}} = \{\mathbf{x}(n), \mathbf{t}(n), \tilde{\mathbf{y}}(n)\}_{n=1}^N$ where $\tilde{\mathbf{y}}$ is the combination of y_f and $\tilde{\mathbf{y}}_{cf}$ (with $\tilde{y}_\eta = y_f$). The ITE generator, $\mathbf{I}$, can then be optimized using $\tilde{\mathcal{D}}$.

We follow a conditional GAN framework similar to the one set out in Mirza & Osindero (2014) to model the latter of these generators. For the former, we have to use a different discriminator in order to capture the same idea. More specifically, GANITE consists of two blocks: a counterfactual imputation block and an ITE block, each of which consists of a generator and a discriminator. We describe each of these blocks and their components in more detail in the following subsection.

3.2 A Detailed Breakdown

Counterfactual generator ($\mathbf{G}$): The counterfactual generator, $\mathbf{G}$, uses the feature vector, $\mathbf{x}$, the treatment vector, $\mathbf{t}$, and the factual outcome, y_f, to generate a potential outcome vector, $\tilde{\mathbf{y}}$. We let g be a function $g : \mathcal{X} \times \{0,1\}^k \times \mathcal{Y} \times [-1,1]^{k-1} \to \mathcal{Y}^k$ and $\mathbf{z_G} \sim \mathcal{U}((-1,1)^{k-1})$. We then define the random variable $\mathbf{G}(\mathbf{x}, \mathbf{t}, y_f)$ as

$$\mathbf{G}(\mathbf{x}, \mathbf{t}, y_f) = g(\mathbf{x}, \mathbf{t}, y_f, \mathbf{z_G}) \tag{3}$$

The goal now is to find a function, g, such that $\mathbf{G}(\mathbf{x}, \mathbf{t}, y_f) \sim \mu_{\mathbf{Y}}(\mathbf{x}, \mathbf{t}, y_f)$. We write $\tilde{\mathbf{y}}$ to denote a sample of $\mathbf{G}$ and $\bar{\mathbf{y}}$ to denote the vector obtained by replacing $\tilde{y}_\eta$ with y_f. Observe that the η-th component of a sample from $\mu_{\mathbf{Y}}(\mathbf{x}, \mathbf{t}, y_f)$ will be y_f, since we are sampling $\mathbf{Y}$ *conditional* on $Y_\eta = y_f$.

Counterfactual discriminator ($\mathbf{D_G}$): We introduce a discriminator, $\mathbf{D_G}$, which maps pairs $(\mathbf{x}, \bar{\mathbf{y}})$ to vectors in $[0,1]^k$ with the i-th component, written $D_{\mathbf{G}}(\mathbf{x}, \tilde{\mathbf{y}})_i$, representing the probability that the i-th component of $\tilde{\mathbf{y}}$ is the factual outcome, equivalently the probability that $\eta = i$. This is in contrast to the standard GAN framework in which the discriminator is given a single sample from one of two distributions and it attempts to determine which distribution it came from. Here the discriminator is given a sample consisting of components from two different distributions and attempts to determine which components came from which distribution.

We train $\mathbf{D_G}$ to maximize the probability of correctly identifying η. We then train $\mathbf{G}$ to maximize the probability of $\mathbf{D_G}$ *incorrectly* identifying η (equivalently we try to minimize the probability of a correct identification - this is the adversarial method of learning between $\mathbf{G}$ and $\mathbf{D_G}$).

Following the framework in Goodfellow et al. (2014), we note that this formulation is captured by a minimax problem given by

$$\min_{\mathbf{G}} \max_{\mathbf{D_G}} \mathbb{E}_{(\mathbf{x},\mathbf{t},y_f)\sim\mu_f} \left[\mathbb{E}_{\mathbf{z_G}\sim\mathcal{U}((-1,1)^k)} \left[\mathbf{t}^T \log \mathbf{D_G}(\mathbf{x},\tilde{\mathbf{y}}) + (\mathbf{1}-\mathbf{t})^T \log(1-\mathbf{D_G}(\mathbf{x},\tilde{\mathbf{y}}))\right]\right] \quad (4)$$

where $\log$ is performed element-wise and ^{T} denotes the transpose operator.

After training the counterfactual generator, we use it to generate the dataset $\tilde{\mathbf{D}}$ and pass this dataset to the ITE block.

ITE generator (I): The ITE generator, $\mathbf{I}$, uses only the feature vector, $\mathbf{x}$, to generate a potential outcome vector, $\hat{\mathbf{y}}$. Similar to our approach with $\mathbf{G}$, let h be a function $h : \mathcal{X} \times [-1,1]^k \to \mathcal{Y}^k$ and $\mathbf{z_I} \sim \mathcal{U}((-1,1)^k)$. We define the random variable $\mathbf{I}(\mathbf{x})$ as

$$\mathbf{I}(\mathbf{x}) = h(\mathbf{x}, \mathbf{z_I}) \quad (5)$$

and similarly, the goal is to find a function, h, such that $\mathbf{I}(\mathbf{x}) \sim \mu_{\mathbf{Y}}(\mathbf{x})$. We write $\hat{\mathbf{y}}$ to denote a sample from $\mathbf{I}(\mathbf{x})$.

ITE discriminator ($\mathbf{D_I}$): Again, we introduce a discriminator, $\mathbf{D_I}$, but this time, since we have access to a *complete* dataset $\tilde{\mathbf{D}}$, we can use a standard conditional GAN discriminator - it takes a pair $(\mathbf{x}, \mathbf{y}^*)$ and returns a scalar corresponding to the probability that $\mathbf{y}^*$ was from the data $\tilde{\mathcal{D}}$ (rather than drawn from $\mathbf{I}$). Again, we train the generator and discriminator in an adversarial fashion using the following minimax criteria

$$\min_{\mathbf{I}} \max_{\mathbf{D_I}} \mathbb{E}_{\mathbf{x}\sim\mu_{\mathbf{X}}} \left[\mathbb{E}_{\mathbf{y}^*\sim\mu_{\mathbf{Y}}}(\mathbf{x}) \left[\log \mathbf{D_I}(\mathbf{x},\mathbf{y}^*)\right] + \mathbb{E}_{\mathbf{y}^*\sim\mathbf{I}(\mathbf{x})} \left[\log(\mathbf{1}-\mathbf{D_I}(\mathbf{x},\mathbf{y}^*))\right]\right] \quad (6)$$

where again $\log$ is taken element-wise.

4 GANITE: OPTIMIZATION

In this section, we describe the empirical loss functions that are used to optimize each component of GANITE. The Pseudo-code is summarized in the Appendix.

4.1 COUNTERFACTUAL BLOCK ($\mathbf{G}, \mathbf{D_G}$):

Based on equation 4, the empirical objective of the minimax problem for $\mathbf{G}$ and $\mathbf{D_G}$ can be defined by

$$V_{CF}(\mathbf{x}(n), \mathbf{t}(n), \bar{\mathbf{y}}(n)) = \mathbf{t}(n)^T \log(\mathbf{D_G}(\mathbf{x}(n), \bar{\mathbf{y}}(n))) + (\mathbf{1}-\mathbf{t}(n))^T \log(\mathbf{1}-\mathbf{D_G}(\mathbf{x}(n), \bar{\mathbf{y}}(n))).$$

We also introduce the following 'supervised' loss in order to enforce the restriction that g_η should be equal to y_f.

$$\mathcal{L}_S^G(y_f(n), \tilde{y}_{\eta(n)}(n)) = (y_f(n) - \tilde{y}_{\eta(n)}(n))^2$$

More specifically, due to the structure of G, it outputs a full vector of potential outcomes, and so it not only outputs counterfactuals, but also gives a value for the one factual that was used as input. We account for this by using $\mathcal{L}_S^G$ to force the generated factual outcome to be close to the actually observed factual outcome. This is because, as noted above, conditional on observing y_f, the component of y corresponding to y_f should clearly be equal to y_f.

With the above two objective functions, $\mathbf{G}$ and $\mathbf{D_G}$ are iteratively optimized with k_G minibatches as follows:

$$\min_{\mathbf{D_G}} -\sum_{n=1}^{k_G} V_{CF}(\mathbf{x}(n), \mathbf{t}(n), \bar{\mathbf{y}}(n))$$

$$\min_{\mathbf{G}} \sum_{n=1}^{k_G} \left[V_{CF}(\mathbf{x}(n), \mathbf{t}(n), \bar{\mathbf{y}}(n)) + \alpha \mathcal{L}_S^G(y_f(n), \tilde{y}_{\eta(n)}(n))\right]$$

where $\alpha \geq 0$ is a hyper-parameter.

4.2 ITE BLOCK ($\mathbf{I}, \mathbf{D_I}$):

After training the counterfactual block ($\mathbf{G}, \mathbf{D_G}$), GANITE optimizes the ITE block ($\mathbf{I}, \mathbf{D_I}$). Based on equation 6, the empirical objective of the minimax problem for $\mathbf{I}$ and $\mathbf{D_I}$ can be defined by

$$V_{ITE}(\mathbf{x}(n), \bar{\mathbf{y}}(n), \hat{\mathbf{y}}(n)) = \log(\mathbf{D_I}(\mathbf{x}(n), \bar{\mathbf{y}}(n))) + \log(1 - \mathbf{D_I}(\mathbf{x}(n), \hat{\mathbf{y}}(n))).$$

Furthermore, in order to optimize the performance with respect to equations 1 and 2, we additionally introduce supervised losses (for the respective cases of $k = 2$ (binary treatments) and $k > 2$ (multiple treatments)) that are defined as follows:

$$(k = 2) : \mathcal{L}_S^I(\bar{\mathbf{y}}(n), \hat{\mathbf{y}}(n)) = ((\bar{y}_1(n) - \bar{y}_0(n)) - (\hat{y}_1(n) - \hat{y}_0(n)))^2$$
$$(k > 2) : \mathcal{L}_S^I(\bar{\mathbf{y}}(n), \hat{\mathbf{y}}(n)) = ||\bar{\mathbf{y}}(n) - \hat{\mathbf{y}}(n)||_2^2.$$

$\mathbf{I}$ and $\mathbf{D_I}$ are then iteratively optimized with k_I minibatches as follows:

$$\min_{\mathbf{D_I}} - \sum_{n=1}^{k_I} V_{ITE}(\mathbf{x}(n), \bar{\mathbf{y}}(n), \hat{\mathbf{y}}(n))$$
$$\min_{\mathbf{I}} \sum_{n=1}^{k_G} \Big[V_{ITE}(\mathbf{x}(n), \bar{\mathbf{y}}(n), \hat{\mathbf{y}}(n)) + \beta \mathcal{L}_S^I(\bar{\mathbf{y}}(n), \hat{\mathbf{y}}(n)) \Big]$$

where $\beta \geq 0$ is a hyper-parameter.

Empirical justification for the inclusion of all the above losses can be found in Section 5.3 where we explore the effect of training with and without each of the losses, see Table 6. We demonstrate there that using a combination of both (for both $\mathbf{G}$ and $\mathbf{I}$) gives the best performance. In addition to this, by using a GAN loss for $\mathbf{I}$ we learn the conditional *distribution* of the potential outcomes rather than just the expectations (which would be the case if we only used the supervised loss $\mathcal{L}_S^I$. See Table 1 in Section 5.3). This allows us to capture the uncertainty of the outcomes, which is very important in the medical setting when treatment decisions need to be made by doctors on the basis of these types of estimations.

Due to the lack of ground truth, it is often difficult in causal inference tasks to optimize the hyper-parameters. More specifically, we do not have access to the true loss function (either PEHE or MSE) that we are trying to minimize, and so it is not possible to select hyper-parameters that minimise the *true* loss. One of the advantages of GANITE, however, is that our target loss can be estimated from the generated counterfactuals, unlike other methods such as in Shalit et al. (2017). Therefore, we can directly optimize the hyper-parameters that minimize this estimated PEHE/MSE over the hyper-parameter space - details of our hyper-parameter optimization and the achieved optimal hyper-parameters are illustrated in the Appendix.

5 EXPERIMENTS

5.1 DATASETS

Due to the nature of the problem, it is very difficult to evaluate the performance of the algorithm on real-world datasets - we never have access to the *ground truth*. Previous works, such as Shalit et al. (2017); Louizos et al. (2017), use both semi-synthetic datasets (either the treatments or the potential outcomes are synthesized) and datasets collected from randomized controlled trials (RCT) to evaluate the ITE generator. We use two semi-synthetic datasets, IHDP and Twins, and one real-world dataset, Jobs, to evaluate the performance of GANITE with various state-of-the-art methods. These datasets are the same as the ones used in Shalit et al. (2017); Louizos et al. (2017). Below, we give a detailed explanation of Twins. The details of IHDP and Jobs are well described in Shalit et al. (2017); Hill (2011); Dehejia & Wahba (2002a) and the Appendix.

Twins: This dataset is derived from all births in the USA between 1989-1991 (Almond et al. (2005)). Among these births, we only focus on the twins. We define the treatment $t = 1$ as being the heavier twin (and $t = 0$ as being the lighter twin). The outcome is defined as the 1-year mortality. For each twin-pair we obtained 30 features relating to the parents, the pregnancy and the birth: marital status; race; residence; number of previous births; pregnancy risk factors; quality of care

during pregnancy; and number of gestation weeks prior to birth. We only chose twins weighing less than 2kg and without missing features (list-wise deletion). This creates a complete dataset (without missing data). The final cohort is 11,400 pairs of twins whose mortality rate for the lighter twin is 17.7%, and for the heavier 16.1%. In this setting, for each twin pair we observed both the case $t = 0$ (lighter twin) and $t = 1$ (heavier twin); thus, the ground truth of individualized treatment effect is known in this dataset. In order to simulate an observational study, we selectively observe one of the two twins using the feature information (creating selection bias) as follows: $t|\mathbf{x} \sim \text{Bern}(\text{Sigmoid}(\mathbf{w}^T\mathbf{x} + n))$ where $\mathbf{w}^T \sim \mathcal{U}((-0.1, 0.1)^{30\times 1})$ and $n \sim \mathcal{N}(0, 0.1)$.

5.2 Performance Metrics and Settings

We use four different performance metrics: expected Precision in Estimation of Heterogeneous Effect (PEHE), average treatment effect (ATE) (Hill (2011)), policy risk ($R_{pol}(\pi)$), and average treatment effect on the treated (ATT) (Shalit et al. (2017)). In this subsection, we only provide definitions for PEHE and $R_{pol}(\pi)$. ATE and ATT are explained (and reported) in the Appendix.

If both factual and counterfactual outcomes are generated from a known distribution (so that we are able to compute the expectations of the outcomes, like in the IHDP dataset), and the treatment is binary, the empirical PEHE (ϵ_{PEHE}) can be defined as follows:

$$\epsilon_{PEHE} = \frac{1}{N}\sum_{n=1}^{N}\Big(\mathbb{E}_{(y_1(n),y_0(n))\sim\mu_{\mathbf{Y}}(\mathbf{x}(n))}[y_1(n) - y_0(n)] - [\hat{y}_1(n) - \hat{y}_0(n)]\Big)^2$$

where $y_1(n), y_0(n)$ are treated and controlled outcomes drawn from the ground truth ($\mu_{\mathbf{Y}}(\mathbf{x})$) and $\hat{y}_1(n), \hat{y}_0(n)$ are their estimations.

If both factual and counterfactual outcomes are observed but the underlying distribution is unknown (like in the Twins dataset), $\hat{\epsilon}_{PEHE}$ can be defined as follows:

$$\hat{\epsilon}_{PEHE} = \frac{1}{N}\sum_{n=1}^{N}\Big([y_1(n) - y_0(n)] - [\hat{y}_1(n) - \hat{y}_0(n)]\Big)^2$$

If only factual outcomes are available but the testing set comes from a randomized controlled trial (RCT), such as in the Jobs dataset, Policy risk ($\mathcal{R}_{pol}(\pi)$) can be defined as follows (Shalit et al. (2017)):

$$R_{pol}(\pi) = \frac{1}{N}\sum_{n=1}^{N}\Big[1 - \Big(\sum_{i=1}^{k}\Big[\frac{1}{|\Pi_i \cap T_i \cap E|}\sum_{\mathbf{x}(n)\in\Pi_i\cap T_i\cap E} y_i(n) \times \frac{|\Pi_i \cap E|}{|E|}\Big]\Big)\Big]$$

where $\Pi_i = \{\mathbf{x}(n) : i = \arg\max \hat{\mathbf{y}}\}$, $T_i = \{\mathbf{x}(n) : t_i(n) = 1\}$, and E is the subset of RCT.

Each dataset is divided 56/24/20% into training/validation/testing sets. Hyper-parameters such as the number of hidden layers α and β are chosen using Random Search (Bergstra & Bengio (2012)). Details about the hyper-parameters are discussed in the Appendix. We run each algorithm 100 times (except for the IHDP dataset, on which we run each algorithm 1,000 times which is the same setting in Shalit et al. (2017)) with new training/validation/testing splits and report the mean and standard deviation of the performances.

5.3 Experimental Results

In our first simulation we focus on demonstrating the effect that including each of the losses introduced in Section 4 has on the performance of the algorithm. As is demonstrated below, inclusion of all four losses gives the best results.

We generate a synthetic dataset as follows: we draw 10,000 10-dimensional feature vectors $\mathbf{x} \sim \mathcal{N}(\mathbf{0}^{10\times 1}, 0.5\times(\Sigma+\Sigma^T))$ where $\Sigma \sim \mathcal{U}((-1, 1)^{10\times 10})$. The treatment assignment is then generated as $t|\mathbf{x} \sim \text{Bern}(\text{Sigmoid}(\mathbf{w}_t^T\mathbf{x} + n_t))$ where $\mathbf{w}_t^T \sim \mathcal{U}((-0.1, 0.1)^{10\times 1})$ and $n_t \sim \mathcal{N}(0, 0.1)$. The potential outcome vector is then generated as $\mathbf{y}|\mathbf{x} \sim (\mathbf{w}_y^T\mathbf{x} + \mathbf{n}_y))$ where $\mathbf{w}_y^T \sim \mathcal{U}((-1, 1)^{10\times 2})$ and $\mathbf{n}_y \sim \mathcal{N}(\mathbf{0}^{2\times 1}, 0.1 \times I^{2\times 2})$. We use 8,000 instances for training and 2,000 instances for testing. We repeat this 100 times and report the average ϵ_{PEHE} on the testing set.

		G		
	PEHE	**S loss only**	**GAN loss only**	**S and GAN loss**
I	**S loss only**	.397 ± .011 (15.6%)	.610 ± .017 (45.1%)	.352 ± .012 (4.8%)
	GAN loss only	.607 ± .044 (44.8%)	.513 ± .029 (34.7%)	.463 ± .015 (27.6%)
	S and GAN loss	.362 ± .011 (7.5%)	.491 ± .030 (31.8%)	.335 ± **.011** (-)

Table 1: Performance with various combinations of GANITE (**G**: Counterfactual generator, **I**: ITE generator, S loss: Supervised loss). Details of each cell are illustrated in the Appendix as Figures.

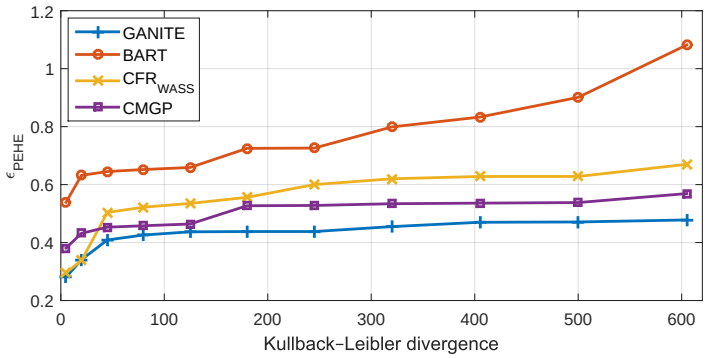


Figure 2: Performance comparison between GANITE and state-of-the-art methods as the selection bias is varied (Kullback-Leibler divergence of treated with respect to controlled distributions)

Table 1 shows the performance of the GANITE architecture using different combinations of the four losses in Section 4. For each generator component, we can use 3 different combinations of loss: (1) Supervised loss (S loss) only (this reduces the corresponding component to a standard neural network), (2) GAN loss only, (3) both S loss and GAN loss. The top-left most entry corresponds to simply using a standard neural network to first impute the counterfactuals and then using another standard neural network to learn an ITE estimator from the imputed dataset. As can be seen, this already performs well, but by adding the GAN losses for both the imputation step and the estimation step, a significant gain is shown (15.6%) (the bottom-right entry).

In Fig. 2 we show that GANITE is robust to an increased selection bias. We generate 10,000 10-dimensional treated samples from $\mathbf{x}_1 \sim \mathcal{N}(\mu_\mathbf{1}, 0.5 \times (\Sigma + \Sigma^T))$ and controlled samples from $\mathbf{x}_0 \sim \mathcal{N}(\mu_\mathbf{0}, 0.5 \times (\Sigma + \Sigma^T))$ where $\Sigma \sim \mathcal{U}((-1, 1)^{10\times 10})$. Fixing $\mu_\mathbf{0}$ and varying $\mu_\mathbf{1}$, we generate various datasets with different Kullback-Leibler divergences (KL divergences) of μ_1 with respect to μ_0. A higher KL divergence indicates a higher selection bias (a larger mismatch) between treated and controlled distributions. As seen in Fig. 2, GANITE robustly outperforms state-of-the-art methods such as Shalit et al. (2017); Alaa & van der Schaar (2017) across the entire range of tested divergences.

Binary treatments: In this section, we evaluate GANITE for estimating individualized treatment effects for binary treatments. We use three datasets and report the ϵ_{PEHE} both in-sample and out-of-sample (for ATE and ATT see the appendix). We compare GANITE with least squares regression using treatment as a feature (OLS/LR$_1$), separate least squares regressions for each treatment (OLS/LR$_2$), balancing linear regression (BLR) (Johansson et al. (2016)), k-nearest neighbor (k-NN) (Crump et al. (2008)), Bayesian additive regression trees (BART) (Chipman et al. (2010)), random forests (RForest) (Breiman (2001)), causal forests (C Forest) (Wager & Athey (2017)), balancing neural network (BNN) (Johansson et al. (2016)), treatment-agnostic representation network (TARNET) (Shalit et al. (2017)), counterfactual regression with Wasserstein distance (CFR$_{WASS}$) (Shalit et al. (2017)), and multi-task gaussian process (CMGP) (Alaa & van der Schaar (2017)). We evaluate both in-sample and out-of-sample performance in Table 2.

Methods	Datasets (Mean ± Std)					
	IHDP ($\sqrt{\epsilon_{PEHE}}$)		**Twins** ($\sqrt{\hat{\epsilon}_{PEHE}}$)		**Jobs** ($\mathcal{R}_{pol}(\pi)$)	
	In-sample	Out-sample	In-sample	Out-sample	In-sample	Out-sample
GANITE	1.9 ± .4	2.4 ± .4	**.289 ± .005**	**.297 ± .016**	**.13 ± .01**	**.14 ± .01**
OLS/LR$_1$	5.8 ± .3*	5.8 ± .3*	.319 ± .001*	.318 ± .007	.22 ± .00*	.23 ± .02*
OLS/LR$_2$	2.4 ± .1	2.5 ± .1	.320 ± .002	.320 ± .003*	.21 ± .00*	.24 ± .01*
BLR	5.8 ± .3*	5.8 ± .3*	.312 ± .003*	.323 ± .018	.22 ± .01*	.25 ± .02*
k-NN	2.1 ± .1	4.1 ± .2*	.333 ± .001*	.345 ± .007*	.02 ± .00	.26 ± .02*
BART	2.1 ± .1	2.3 ± .1	.347 ± .009*	.338 ± .016	.23 ± .00*	.25 ± .02*
R Forest	4.2 ± .2*	6.6 ± .3*	.306 ± .002*	.321 ± .005	.23 ± .01*	.28 ± .02*
C Forest	3.8 ± .2*	3.8 ± .2*	.366 ± .003*	.316 ± .011	.19 ± .00*	.20 ± .02*
BNN	2.2 ± .1	2.1 ± .1	.325 ± .003*	.321 ± .018	.20 ± .01*	.24 ± .02*
TARNET	.88 ± .02	.95 ± .02	.317 ± .005*	.315 ± .003	.17 ± .01*	.21 ± .01*
CFR$_{WASS}$	.71 ± .02	**.76 ± .02**	.315 ± .007*	.313 ± .008	.17 ± .01*	.21 ± .01*
CMGP	**.65 ± .44**	.77 ± .11	.320 ± .002*	.319 ± .008	.22 ± .03*	.24 ± .05

Table 2: Performance of ITE estimation with three real-world datasets. Bold indicates the method with the best performance for each dataset. *: is used to indicate methods that GANITE shows a statistically significant improvement over.

As can be seen in Table 2, GANITE achieves significant performance gains on the Twins and Jobs datasets in comparison with state-of-the-art methods (both in-sample and out-of-sample [2]). GANITE achieves a much higher gain for individualized treatment effect estimations (such as $\hat{\epsilon}_{PEHE}$ and $\mathcal{R}_{pol}(\pi)$) than average treatment effect estimations (such as $\hat{\epsilon}_{ATE}$ and ϵ_{ATT}). On IHDP, GANITE is competitive with BART and BNN but is outperformed by TARNET, CFR$_{WASS}$ and CMGP. We believe this is due to the fact that GANITE has a large number of parameters to be optimized and IHDP is a relatively small dataset (747 samples). This belief is backed up by our significant gains over these methods in both Twins and Jobs, where the number of samples is much larger, (11400 and 3212 samples, respectively).

Methods	**Metric**: MSE$_y$			
	In Sample	**Gain** (%)	**Out Sample**	**Gain** (%)
GANITE	**.0427 ± .0161**	(-)	**.0723 ± .0183**	(-)
OLS/LR$_1$	.0855 ± .0096	50.1%	.0871 ± .0142	17.0%
OLS/LR$_2$	.0857 ± .0099	50.2%	.0883 ± .0147	18.1%
BLR	.0996 ± .0081*	57.1%	.1017 ± .0127	28.9%
KNN	.0930 ± .0101*	54.1%	.1008 ± .0236	28.3%
BART	.1097 ± .0084*	61.1%	.1037 ± .0283	30.3%
R Forest	.0442 ± .0069	3.4%	.0927 ± .0138	22.0%
C Forest	.1607 ± .0014*	73.4%	.1665 ± .0035*	56.6%
BNN	.0602 ± .0102	29.1%	.1031 ± .0145	29.9%
TARNET	.0854 ± .0091	50.0%	.0879 ± .0030	17.7%
CFR$_{WASS}$	.0896 ± .0036*	52.3%	.0894 ± .0057	19.1%
CMGP	.0844 ± .0073*	49.4%	.0793 ± .0191	8.3%

Table 3: Performance of multiple treatment effects estimations using Twins data. Bold indicates the method with the best performance for each dataset. *: is used to indicate methods that GANITE shows a statistically significant improvement over.

[2]Note that both "in-sample" and "out-sample" experiments evaluate the performance of **I** networks.

Multiple treatments: GANITE is naturally defined for estimating multiple treatment effects. In this subsection, we further preprocess the Twins data to create a dataset containing multiple treatments. The multiple treatments are determined as follows: (1) $t = 1$: lower weight, female sex, (2) $t = 2$: lower weight, male sex, (3) $t = 3$: higher weight, female sex, (4) $t = 4$: higher weight, male sex. Therefore, we have 4 possible treatments for each sample.

We use the mean-squared error:

$$\text{MSE}_y = \frac{1}{N \times |\mathcal{T}_i|} \sum_{i=1}^{N} \sum_{t \in \mathcal{T}_i} \Big(y_t(x_i) - \hat{y}_t(x_i) \Big)^2$$

as the performance metric to evaluate the multiple treatment effects (other metrics, such as PEHE, do not have a natural extension to the multiple treatments setting). For comparison with state-of-the-art methods, we naively extend BLR, C Forest, BNN, TARNET, CFR$_{WASS}$, and CMGP for multiple treatments: one of the four treatments is selected as a control treatment and then the remaining three create three separate binary ITE estimation problems (all against the same chosen control treatment).

As can be seen in Table 3 compared with Table 2 GANITE significantly outperforms other state-of-the-art methods such as TARNET and CFR$_{WASS}$ (17.7% and 19.1% gains in terms of out sample MSE, respectively). This is because, GANITE is designed for multiple treatments; the model is jointly trained for all treatments. On the other hand, other methods are designed for binary treatments and only naively extend to multiple treatments by training pairs of the available treatments.

5.4 Discussion

The experimental results provide various intuitions of the GANITE framework for ITE estimation. First, GANITE can be easily extended to any number of treatments and performs well in this multiple treatment setting. As can be seen in Section 4, however, a different loss for binary treatment and multiple treatments must be used because PEHE is only defined for the binary treatment setting - the MSE is not a natural generalisation of the PEHE and we believe exploration of possible loss functions in this setting would be an interesting future work.

A further extension to this problem would be to consider a setting in which a patient may receive several treatments (rather than just one). While this work can handle this problem naively (by treating each *combination* of treatments as a separate 'treatment') we believe this would also be an interesting problem to explore in a future work.

6 Conclusion

In this paper we introduced a novel method for dealing with the ITE estimation problem. We have shown empirically that our method is more robust to large selection biases and performs better on standard benchmark datasets than other state-of-the-art methods. Our method also achieves significant performance gains over state-of-the-art when estimating ITE for multiple treatments because it is able to jointly estimate the representations across the multiple treatments.

Acknowledgments

This work was supported by the Office of Naval Research (ONR) and the NSF (Grant number: ECCS1462245, ECCS1533983, and ECCS1407712).

REFERENCES

Ahmed M Alaa and Mihaela van der Schaar. Bayesian inference of individualized treatment effects using multi-task gaussian processes. *NIPS*, 2017.

Ahmed M Alaa, Michael Weisz, and Mihaela van der Schaar. Deep counterfactual networks with propensity-dropout. *ICML Workshop on Principled Approaches to Deep Learning*, 2017.

Douglas Almond, Kenneth Y Chay, and David S Lee. The costs of low birth weight. *The Quarterly Journal of Economics*, 120(3):1031–1083, 2005.

Susan Athey and Guido Imbens. Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360, 2016.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Hugh A Chipman, Edward I George, Robert E McCulloch, et al. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

Richard K Crump, V Joseph Hotz, Guido W Imbens, and Oscar A Mitnik. Nonparametric tests for treatment effect heterogeneity. *The Review of Economics and Statistics*, 90(3):389–405, 2008.

Rajeev H Dehejia and Sadek Wahba. Propensity score-matching methods for nonexperimental causal studies. *The review of economics and statistics*, 84(1):151–161, 2002a.

Rajeev H Dehejia and Sadek Wahba. Propensity score-matching methods for nonexperimental causal studies. *The review of economics and statistics*, 84(1):151–161, 2002b.

Vincent Dorie. Npci: Non-parametrics for causal inference, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural information processing systems*, pp. 2672–2680, 2014.

A Hannart, J Pearl, FEL Otto, P Naveau, and M Ghil. Causal counterfactual theory for the attribution of weather and climate-related events. *Bulletin of the American Meteorological Society*, 97(1): 99–110, 2016.

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

M Höfler. Causal inference based on counterfactuals. *BMC medical research methodology*, 5(1):28, 2005.

Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *International Conference on Machine Learning*, pp. 3020–3029, 2016.

James K Kirklin, David C Naftel, Robert L Kormos, Lynne W Stevenson, Francis D Pagani, Marissa A Miller, Karen L Ulisney, J Timothy Baldwin, and James B Young. Second intermacs annual report: more than 1000 primary lvad implants. *The Journal of heart and lung transplantation: the official publication of the International Society for Heart Transplantation*, 29(1):1, 2010.

Robert J LaLonde. Evaluating the econometric evaluations of training programs with experimental data. *The American economic review*, pp. 604–620, 1986.

Christos Louizos, Uri Shalit, Joris Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep latent-variable models. *NIPS*, 2017.

Min Lu, Saad Sadiq, Daniel J Feaster, and Hemant Ishwaran. Estimating individual treatment effect in observational data using random forest methods. *arXiv preprint arXiv:1701.05306*, 2017.

Jared K Lunceford and Marie Davidian. Stratification and weighting via the propensity score in estimation of causal treatment effects: a comparative study. *Statistics in medicine*, 23(19):2937–2960, 2004.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Kristin E Porter, Susan Gruber, Mark J Van Der Laan, and Jasjeet S Sekhon. The relative performance of targeted maximum likelihood estimators. *The International Journal of Biostatistics*, 7 (1):1–34, 2011.

Donald B Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

Uri Shalit, Fredrik Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. *ICML*, 2017.

Jeffrey A Smith and Petra E Todd. Does matching overcome lalonde's critique of nonexperimental estimators? *Journal of econometrics*, 125(1):305–353, 2005.

Peter Spirtes. A tutorial on causal inference. 2009.

Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, (just-accepted), 2017.

APPENDIX

PSEUDO-CODE OF GANITE

Algorithm 1 Pseudo-code of GANITE

while convergence of training loss of $\mathbf{G}$ and $\mathbf{D_G}$ **do**

(1) Counterfactual block optimization

Use k_C minibatches, iteratively optimize $\mathbf{G}, \mathbf{D_G}$ by stochastic gradient descent (SGD)

$$\min_{\mathbf{D_G}} -\sum_{n=1}^{k_G} V_{CF}(\mathbf{x}(n), \mathbf{t}(n), \tilde{\mathbf{y}}(n))$$

$$\min_{\mathbf{G}} \sum_{n=1}^{k_G} \Big[V_{CF}(\mathbf{x}(n), \mathbf{t}(n), \tilde{\mathbf{y}}(n)) + \alpha \mathcal{L}_S^G (y_f(n), \tilde{y}^*_{\eta(n)}(n)) \Big]$$

while convergence of training loss of $\mathbf{I}$ and $\mathbf{D_I}$ **do**

(2) ITE block optimization

Use k_I minibatches, update $\mathbf{I}, \mathbf{D_I}$ by SGD

$$\min_{\mathbf{D_I}} -\sum_{n=1}^{k_I} V_{ITE}(\mathbf{x}(n), \tilde{\mathbf{y}}(n), \hat{\mathbf{y}}(n))$$

$$\min_{\mathbf{I}} \sum_{n=1}^{k_G} \Big[V_{ITE}(\mathbf{x}(n), \tilde{\mathbf{y}}(n), \hat{\mathbf{y}}(n)) + \beta \mathcal{L}_S^I (\tilde{\mathbf{y}}(n), \hat{\mathbf{y}}(n)) \Big]$$

DETAILED DESCRIPTION OF THE DATASETS

IHDP

Hill (2011) provided a dataset for ITE estimation with the Infant Health and Development Program (IHDP). The dataset consists of 747 children ($t = 1$: 139, $t = 0$ 608) with 25 features. We generated potential outcomes from setting A in the NPCI package Dorie (2016).

JOBS

Jobs data studied in LaLonde (1986) is composed of randomized data based on the National Supported Work program and non-randomized data from observational studies. We use a (random) subset of the randomized data to evaluate the algorithms based on $\mathcal{R}_{pol}(\pi)$ and ϵ_{ATT}. The dataset consists of 722 randomized samples ($t = 1$: 297, $t = 0$: 425) and 2490 non-randomized samples ($t = 1$: 0, $t = 0$: 2490), all with 7 features.

SUMMARY OF THE DATASETS

Data	**Condition**					**Property**	
	F	**CF**	**Distribution**	**RT-test**	T	N	s
IHDP	✓	✓	Known		Binary	747	25
Jobs	✓	X	Unknown	✓	Binary	3212	7
Twins-Binary	✓	✓	Unknown		Binary	11400	30
Twins-Multiple	✓	X	Unknown		Multiple	11400	30

Table 4: Summary of the datasets (N is the number of samples, s is the feature-dimension)

Methods	Datasets (Mean ± Std)					
	IHDP (ϵ_{ATE})		Twins ($\hat{\epsilon}_{ATE}$)		Jobs (ϵ_{ATT})	
	In-sample	Out-sample	In-sample	Out-sample	In-sample	Out-sample
GANITE	.43 ± .05	.49 ± .05	.0058 ± .0017	.0089± 0.0075	**.01 ± .01**	**.06 ± .03**
OLS/LR$_1$	.73 ± .04*	.94 ± .06*	**.0038 ± .0025**	.0069 ± .0056	**.01 ± .00**	.08 ± .04
OLS/LR$_2$	.14 ± .01	.31 ± .02	.0039 ± .0025	.0070 ± .0059	**.01 ± .01**	.08 ± .03
BLR	.72 ± .04*	.93 ± .05*	.0057 ± .0036	.0334 ± .0092*	**.01 ± .01**	.08 ± .03
k-NN	.14 ± .01	.90 ± .05*	.0028 ± .0021	**.0051 ± .0039**	.21 ± .01*	.13 ± .05
BART	.23 ± .01	.34 ± .02	.1206 ± .0236*	.1265 ± .0234*	.02 ± .00	.08 ± .03
R Forest	.73 ± .05*	.96 ± .06*	.0049 ± .0034	.0080 ± .0051	.03 ± .01	.09 ± .04
C Forest	.18 ± .01	.40 ± .03	.0286 ± .0035*	.0335 ± .0083*	.03 ± .01	.07 ± .03
BNN	.37 ± .03	.42 ± .03	.0056 ± .0032	.0203 ± .0071	.04 ± .01	.09 ± .04
TARNET	.26 ± .01	.28 ± .01	.0108 ± .0017*	.0151 ± .0018	.05 ± .02	.11 ± .04
CFR$_{WASS}$	.25 ± .01	.27 ± .01	.0112 ± .0016*	.0284 ± .0032*	.04 ± .01	.09 ± .03
CMGP	**.11 ± .10**	**.13 ± .12**	.0124 ± .0051	.0143 ± .0116	.06 ± .06	.09 ± .07

Table 5: Performance of average treatment effect estimation. Bold represents the best performance. *: statistically significant improvement of GANITE.

PERFORMANCE METRICS AND THE RESULTS OF AVERAGE TREATMENT EFFECT ESTIMATION

In this subsection, we use two different performance metrics for average treatment effect (ATE) estimation: average treatment effect (ATE) (Hill (2011)), and average treatment effect on the treated (ATT) (Shalit et al. (2017)).

If both factual and counterfactual outcomes are generated from a known distribution (and so we are able to compute the expected value, such as in IHDP), the error of ATE (ϵ_{ATE}) is defined as:

$$\epsilon_{ATE} = ||\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{\mathbf{y}(n)\sim\mu_{\mathbf{Y}}(\mathbf{x}(n))}[\mathbf{y}(n)] - \frac{1}{N}\sum_{i=1}^{n}\hat{\mathbf{y}}(n)||_2^2$$

where $\hat{\mathbf{y}}$ is the estimated potential outcome.

If both factual and counterfactual outcomes are observed but the underlying distribution is unknown (like in Twins), $\hat{\epsilon_{ATE}}$ is defined as:

$$\hat{\epsilon}_{ATE} = ||\frac{1}{N}\sum_{n=1}^{N}\mathbf{y}(n) - \frac{1}{N}\sum_{n=1}^{N}\hat{\mathbf{y}}(n)||_2^2$$

If only factual outcomes are available (such as in Jobs), treatment is binary, and the testing set comes from a randomized controlled trial (RCT), the true average treatment effect on the treated (ATT) is defined as follows (Shalit et al. (2017)):

$$ATT = \frac{1}{|T_1 \cap E|}\sum_{\mathbf{x}_i \in T_1 \cap E} Y_1(\mathbf{x}_i) - \frac{1}{|T_0 \cap E|}\sum_{\mathbf{x}_i \in C \cap E} Y_0(\mathbf{x}_i)$$

$$\epsilon_{ATT} = |ATT - \frac{1}{|T_1 \cap E|}\sum_{\mathbf{x}_i \in T_1 \cap E} \hat{Y}_1(\mathbf{x}_i) - \hat{Y}_0(\mathbf{x}_i)|$$

where T_1 is the subset corresponding to treated samples, T_0 is the subset corresponding to controlled samples, and E is the subset corresponding to the randomized controlled trials.

Table. 5 shows the performance of the various algorithms with respect to these metrics.

As can be seen in Table 5, the GANITE achieves competitive performances for Average Treatment Effect (ATE) estimation but not the best model to estimate ATE (except the Jobs dataset). However, we do not believe that it is an important metric for distinguishing models where the task is predicting

treatment effects on an individual level. The problem we address with GANITE is to estimate the ITE. We used the ATE performance as a sanity check for our method - and believe it passes the sanity check, being competitive with most other methods.

HYPER-PARAMETER OPTIMIZATION

We optimize our hyper-parameters in GANITE by estimating the PEHE (MSE in the case of multiple treatments) on the dataset generated by G and minimizing this with respect to the hyper-parameters. The table below indicates specifics of this process, including the values we search over.

Table 6: Hyper-parameters of GANITE

Blocks	**Sets of Hyper-parameters**
Initialization	Xavier Initialization for Weight matrix, Zero initialization for bias vector.
Optimization	Adam Moment Optimization
Batch size (k_G, k_I)	$\{32, 64, 128, 256\}$
Depth of layers	$\{1, 3, 5, 7, 9\}$
Hidden state dimension	{s, int(s/2), int(s/3), int(s/4), int(s/5)}
α, β	$\{0, 0.1, 0.5, 1, 2, 5, 10\}$

For the hyper-parameter optimization of the benchmarks, we follow the hyper-parameter optimzation code published in the github with their main code. For instance, the hyper-parameters of CFR$_{WASS}$ are optimized using `cfr_param_search.py` file which is published in `https://github.com/clinicalml/cfrnet`

OPTIMAL HYPER-PARAMETERS FOR EACH DATASET

Table 7: Optimal Hyper-parameters of GANITE

Dataset	**Optimal Hyper-parameters**
IHDP	$k_G : 64, k_I : 64$, Depth of layers: 5, h_{dim}: 8, $\alpha : 2, \beta : 5$
Jobs	$k_G : 128, k_I : 128$, Depth of layers: 3, h_{dim}: 4, $\alpha : 1, \beta : 5$
Twins - Binary	$k_G : 128, k_I : 128$, Depth of layers: 5, h_{dim}: 8, $\alpha : 2, \beta : 2$
Twins - Multiple	$k_G : 128, k_I : 128$, Depth of layers: 7, h_{dim}: 8, $\alpha : 1, \beta : 2$

ADDITIONAL EXPERIMENTS

PERFORMANCE GAP BETWEEN G AND I

As explained in Section 4, I tries to learn the potential distribution that consists of factual outcomes and counterfactual outcomes that are generated by G. To evaluate how well I is able to learn from G, we compare the performance of G and I of the GANITE framework in Table 8 in terms of ITE estimation. As can be seen in Table 8, the in-sample ITE estimation performance of GANITE (I) is competitive with GANITE (G). It experimentally verifies that GANITE (I) learns well from the outputs of GANITE (G).

ZERO OUT THE CONTRIBUTION OF FACTUAL OUTCOME AND TREATMENT

In this section, we use the trained G to compute ITE only with x. The learned function G needs x, t, and y_f as the inputs. Therefore, in order to compute ITE only with x, we should zero out the contribution of t and y_f in the G function. G tries to learn the conditional probability $\mathcal{P}(y|x, y_f, t)$

Methods	Datasets (Mean ± Std)		
	IHDP ($\sqrt{\epsilon_{PEHE}}$)	**Twins** ($\sqrt{\hat{\epsilon}_{PEHE}}$)	**Jobs** ($\mathcal{R}_{pol}(\pi)$)
GANITE (I)	1.9 ± .4	.289 ± .005	.13 ± .01
GANITE (G)	1.4 ± .2	.267 ± .004	.10 ± .01

Table 8: Performance comparison of in-sample ITE estimation between GANITE (I) and GANITE (G) with three real-world datasets.

and what we want to compute is the conditional probability $\mathcal{P}(y|x)$. Therefore, zero out the impact of t and y_f can be done as follows.

$$\mathcal{P}(y|x) = \int \mathcal{P}(y|x,t,y_f)P(y_f,t|x)dtdy_f$$

The $\mathcal{P}(y|x,t,y_f)$ is learned by G and $P(y_f,t|x)$ can be easily learned using supervised learning framework (all the labels (y_f,t) are available) with drop-out approach (to approximate the integral as the sample mean of multiple samples). We use multi-layer perceptron (MLP) with multiple outputs to learn the function $P(y_f,t|x)$. We called this as zero-out GANITE. We compare the performance of zero-out GANITE to original GANITE in Table 9. As can be seen in Table 9, the performance of original GANITE is marginally better than zero-out GANITE in three different datasets. We do not believe that zero-out GANITE would be any simpler than our proposed structure - in both cases we would still have 2 learning stages (Table 9 experimentally verifies this).

Methods	Datasets (Mean ± Std)					
	IHDP ($\sqrt{\epsilon_{PEHE}}$)		**Twins** ($\sqrt{\hat{\epsilon}_{PEHE}}$)		**Jobs** ($\mathcal{R}_{pol}(\pi)$)	
	In-sample	Out-sample	In-sample	Out-sample	In-sample	Out-sample
GANITE	1.9 ± .4	2.4 ± .4	.289 ± .005	.297 ± .016	.13 ± .01	.14 ± .01
zero-out GANITE	2.2 ± .6	2.6 ± .7	.297 ± .008	.308 ± .022	.14 ± .02	.17 ± .01

Table 9: Performance comparison of ITE estimation between GANITE and zero-out GANITE with three real-world datasets.

TOY EXAMPLE

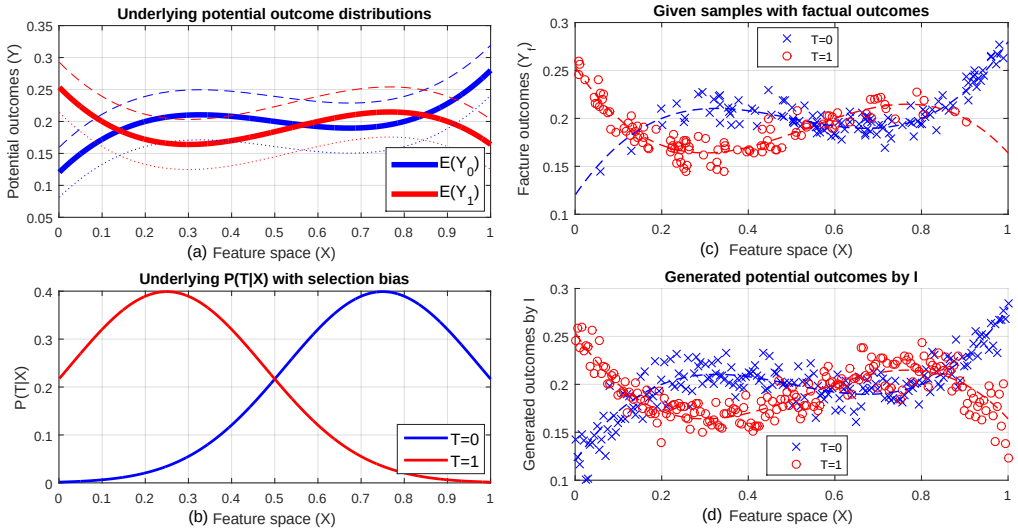
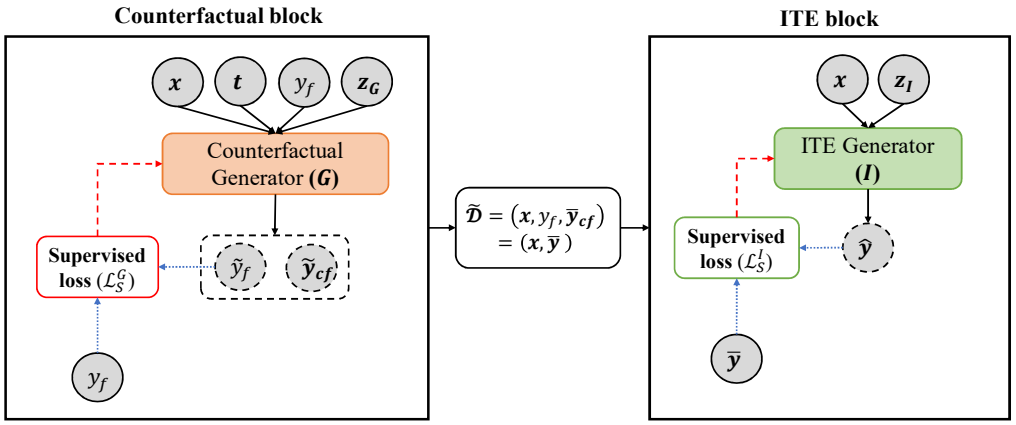
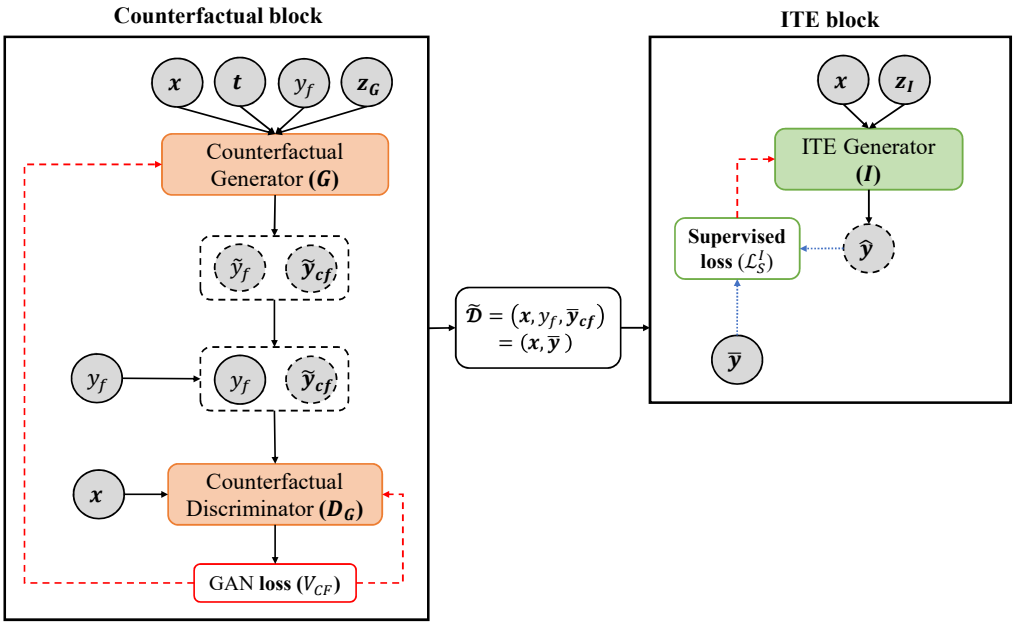


Figure 3: (a) Underlying distribution of potential outcomes ($\mathbf{Y}$), (b) Underlying distribution of treatment assignments $\mathcal{P}(T|\mathbf{X})$, (c) Training data (factual outcomes) sampled from distributions explained in (a) and (b), (d) Potential outcomes sampled from trained ITE generator ($\mathbf{I}$).

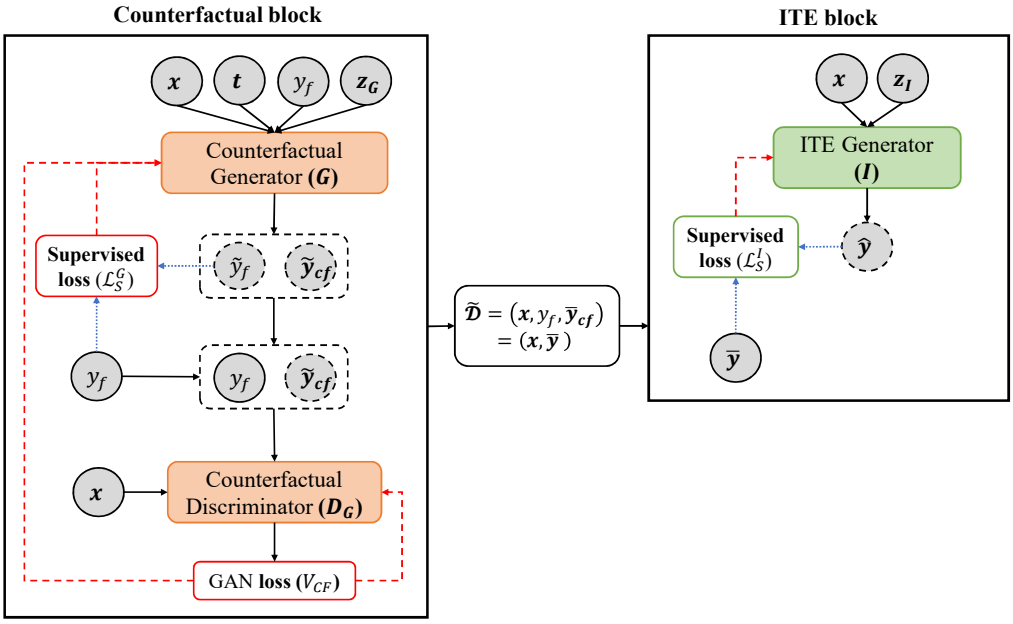
ADDITIONAL FIGURES FOR TABLE 1



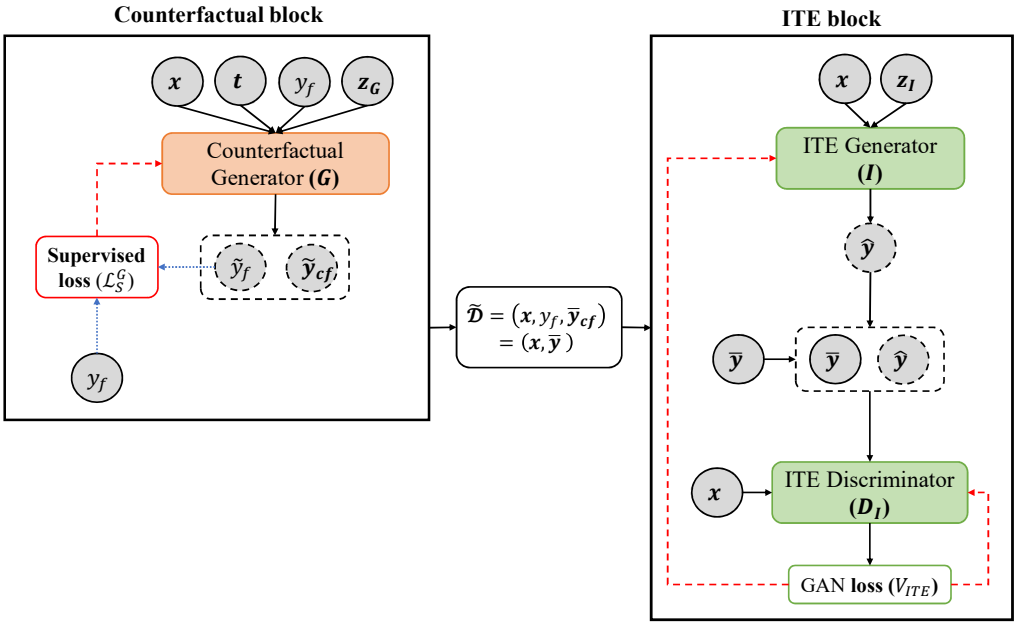
(a) **G:** S loss only, **I:** S loss only



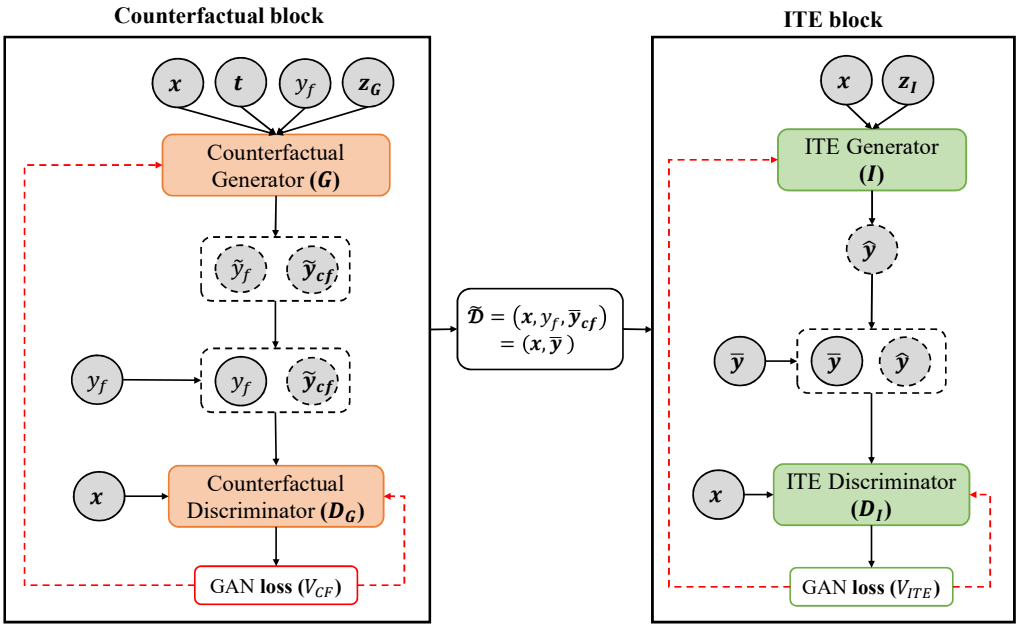
(b) **G:** GAN loss only, **I:** S loss only



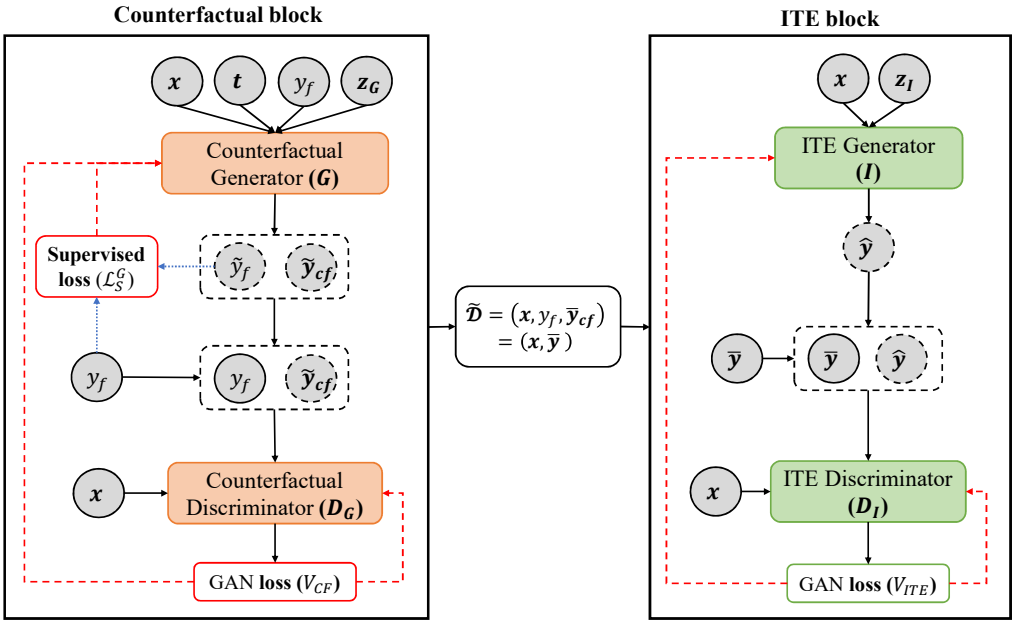
(c) **G:** S and GAN loss**, I:** S loss only



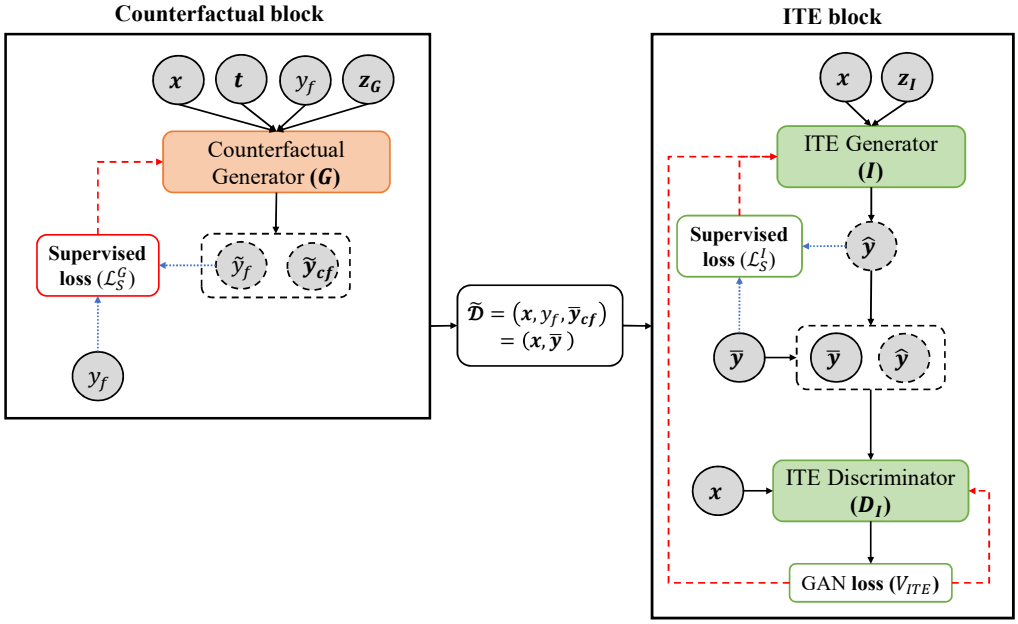
(d) **G:** S loss only**, I:** GAN loss only



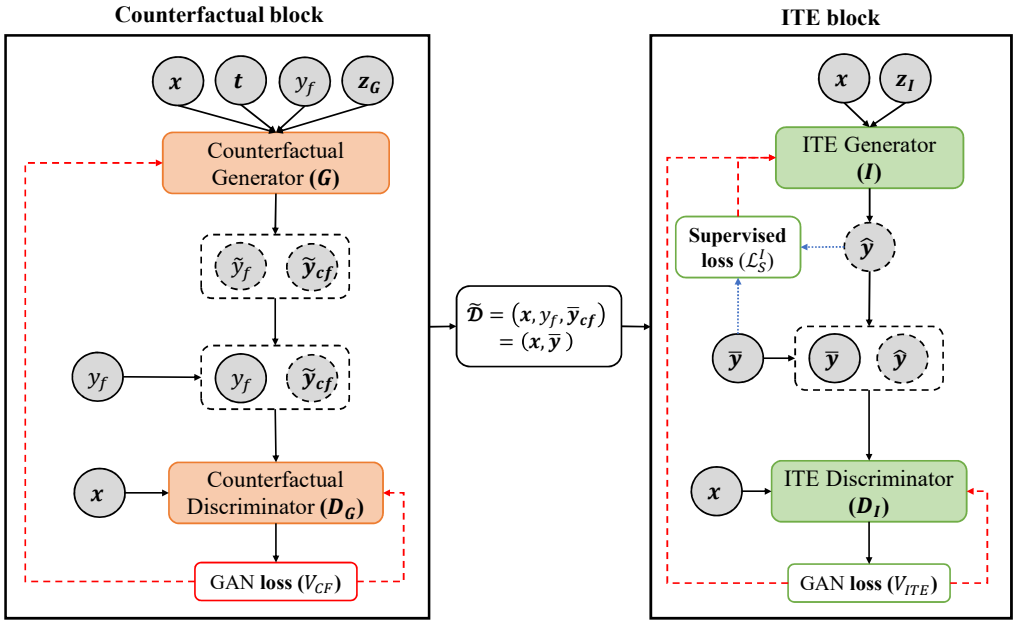
(e) **G:** GAN loss only**, I:** GAN loss only



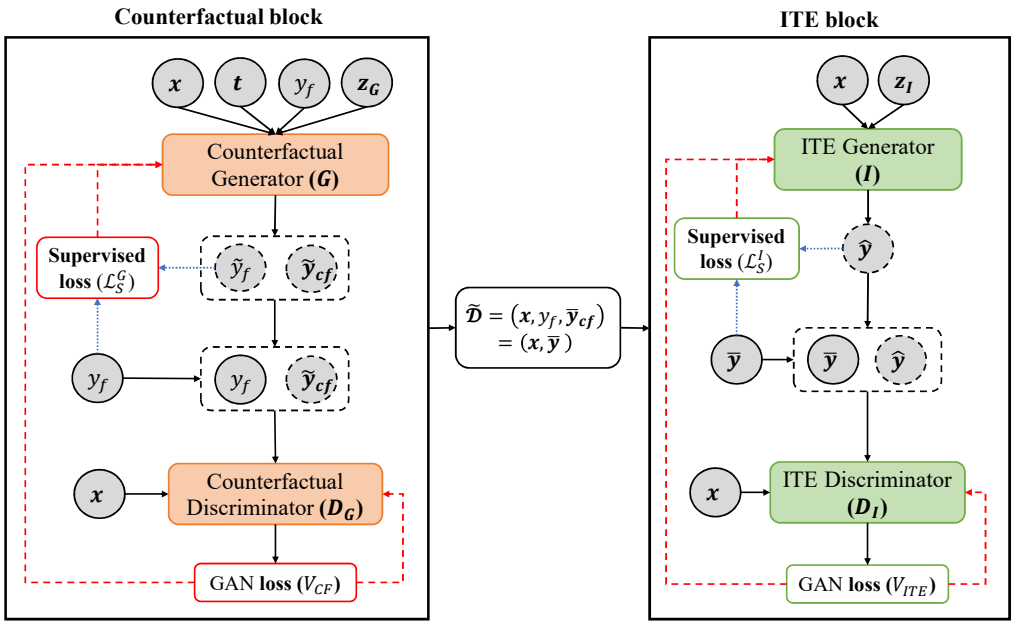
(f) **G:** S and GAN loss**, I:** GAN loss only



(g) **G:** S loss only, **I:** S and GAN loss



(h) **G:** GAN loss only, **I:** S and GAN loss



(i) **G:** S and GAN loss**, I:** S and GAN loss

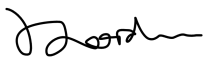
Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	GANITE: Estimation of Individualised Treatment Effects using Generative Adversarial Nets
Publication Status	☑ Published □ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Jinsung Yoon, James Jordon, and Mihaela van der Schaar. GANITE: Estimation of Individualised Treatment Effects using Generative Adversarial Nets. In *International Conference on Learning Representations*, 2018.

Student Confirmation

Student Name:	James Jordon		
Contribution to the Paper	• I wrote significant parts of the paper, including the entire problem formulation and methodological section. Jinsung wrote the first draft of the experiments section, which I rewrote significantly. • I worked with Jinsung to analyse the results, and proposed the ablation study that analyses precisely where the gains in performance come from • All authors contributed to the development of the paper through multiple discussions. • All authors reviewed the final draft.		
Signature 		Date	09/06/2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Professor Mihaela van der Schaar		
Supervisor comments I agree that the candidate has made a substantial contribution to the publication.		
Signature 	Date	09/06/2021

This completed form should be included in the thesis, at the end of the relevant chapter.

arXiv:2002.12326v2 [cs.LG] 22 Nov 2020

Estimating the Effects of Continuous-valued Interventions using Generative Adversarial Networks

Ioana Bica*
Department of Engineering Science
University of Oxford, Oxford, UK
The Alan Turing Institute, London, UK
ioana.bica@eng.ox.ac.uk

James Jordon*
Department of Engineering Science
University of Oxford, Oxford, UK
james.jordon@wolfson.ox.ac.uk

Mihaela van der Schaar
University of Cambridge, Cambridge, UK
University of California, Los Angeles, USA
The Alan Turing Institute, London, UK
mv472@cam.ac.uk

Abstract

While much attention has been given to the problem of estimating the effect of discrete interventions from observational data, relatively little work has been done in the setting of continuous-valued interventions, such as treatments associated with a dosage parameter. In this paper, we tackle this problem by building on a modification of the generative adversarial networks (GANs) framework. Our model, SCIGAN, is flexible and capable of simultaneously estimating counterfactual outcomes for several different continuous interventions. The key idea is to use a significantly modified GAN model to learn to generate counterfactual outcomes, which can then be used to learn an inference model, using standard supervised methods, capable of estimating these counterfactuals for a new sample. To address the challenges presented by shifting to continuous interventions, we propose a novel architecture for our discriminator - we build a hierarchical discriminator that leverages the structure of the continuous intervention setting. Moreover, we provide theoretical results to support our use of the GAN framework and of the hierarchical discriminator. In the experiments section, we introduce a new semi-synthetic data simulation for use in the continuous intervention setting and demonstrate improvements over the existing benchmark models.

1 Introduction

Estimating the personalised effects of interventions is crucial for decision making in many domains such as medicine, education, public policy and advertising. Such domains have a wealth of observational data available. Most of the methods developed in the causal inference literature focus on learning the counterfactual outcomes of discrete interventions, such as binary or categorical treatments[2] [1–9]. Unfortunately, in many cases, deciding how to intervene involves not only deciding which intervention to make (e.g. whether to treat cancer with radiotherapy, chemotherapy or surgery) but also deciding on the value of some continuous parameter associated with intervening (e.g. the dosage of radiotherapy to be administered). In medicine there are many examples of treatments

*Equal contribution.

[2]For ease of exposition, we will sometimes refer to interventions as treatments and to the associated continuous parameter as the dosage throughout the paper.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

that are associated with a continuous dosage parameter (such as vasopressors [10]). In the medical setting, using a high dosage for a treatment can lead to toxic effects while using a low dosage can result in no effect on the patient outcome [11, 12]. In other domains, there are many examples of continuous interventions, such as the duration of an education or job training program, the frequency of an exposure or the price used in an advert. Naturally, being able to estimate the effect of these continuous interventions will aid in the decision making process.

Learning from observational data already presents significant challenges when there is only a single intervention (and thus the decision is binary - whether to intervene or not). As explained in [13], in an observational dataset, only the factual outcome is present - the "counterfactual" outcomes are not observed. This problem is exacerbated in the setting of continuous interventions where the number of counterfactuals is no longer even finite. Moreover, the decision to intervene is non-random and instead is assigned according to the features associated with each sample. Due to the continuous nature of the interventions, adjusting for selection bias is significantly more complex than for binary (or even multiple) interventions. Thus, standard methods for adjusting for selection bias for discrete treatments cannot be easily extended to handle bias in the continuous setting.

We propose SCIGAN (eStimating the effects of Continuous Interventions using GANs). We build on the GAN framework of [14] to learn the distribution of the unobserved counterfactuals. GANs have already been used in GANITE [6] to generate the unobserved counterfactual outcomes for discrete interventions. The intuition is that if a counterfactual generator and discriminator are trained adversarially, then the generator can fool the discriminator (i.e. the discriminator will not be able to correctly identify the factual outcome) by generating counterfactuals according to their true distribution. Unfortunately, no theoretical work was provided in [6] to back up this intuition. A key contribution of this paper is to provide theoretical results that justify using the GAN framework to learn to generate counterfactual outcomes; these results also apply to GANITE.

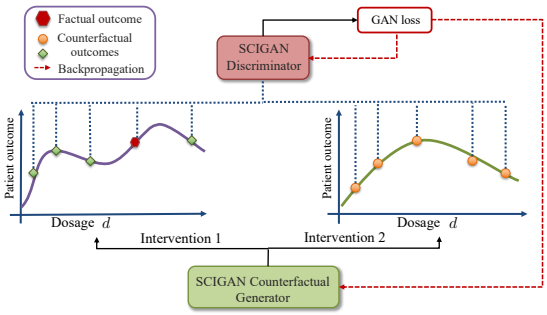


Figure 1: Overview of GAN framework used as part of SCIGAN for learning the distribution of the counterfactual outcomes.

GANITE itself presents a significant modification to the original GAN framework - rather than the discriminator discriminating between entirely real or entirely fake samples, the discriminator is attempting to identify the real component from a vector containing the real (factual) outcome from the dataset and the fake (counterfactual) outcomes generated by the generator. SCIGAN inherits this key difference from a standard GAN. However, beyond our theoretical contribution, we propose significant changes to the generator and discriminator in order to tackle the more complex problem of estimating outcomes of continuous interventions.

We define a discriminator that acts on a finite set of points from each generated response curve (rather than on entire curves), as shown in Fig. 1. We draw on ideas from [15] to ensure that our discriminator acts as a function of a set, rather than of a vector. Moreover, we focus on the setting of interventions *each* with an associated continuous parameter. In this setting, we propose a *hierarchical* discriminator which breaks down the job of the discriminator into determining the factual intervention and determining the factual parameter using separate networks. We show in the experiments section that this approach significantly improves performance and is more stable than using a single network discriminator. We also model the generator as a multi-task deep network capable of taking a continuous parameter as an input; this gives us the flexibility to learn heterogeneous response curves for the different interventions.

Our contributions in this paper are 4-fold: (1) we propose SCIGAN, a significantly modified GAN framework, capable of estimating outcomes for continuous and many-level-discrete interventions, (2) we provide theoretical justification for both the use of a GAN framework and a hierarchical discriminator, (3) we propose novel architectures for each of our networks, (4) we propose a new semi-synthetic data simulation for use in the continuous intervention setting. We show, using semi-synthetic experiments, that our model outperforms existing benchmarks.

2 Related work

Methods for estimating the outcomes of treatments with a continuous dosage from observational data make use of the generalized propensity score (GPS) [16–18] or build on top of balancing methods for multiple treatments. [19] developed a neural network based method to estimate counterfactuals for multiple treatments and continuous dosages. The proposed Dose Response networks (DRNets) in [19] consist of a three level architecture with shared layers for all treatments, multi-task layers for each treatment and additional multi-task layers for dosage sub-intervals. Specifically, for each treatment w, the dosage interval $[a_w, b_w]$ is subdivided into E *equally* sized sub-intervals and a multi-task head is added for each sub-interval. This is an extension of the architecture in [20]. However, the main advantage of using multi-task heads for dosage intervals would be the added flexibility in the model to learn potentially very different functions over different regions of the dosage interval. DRNets do not determine these intervals dynamically and thus much of this flexbility is lost. Our approach (using GANs to generate counterfactuals) fundamentally differs from DRNets (supervised learning with bias-adjustment) and we demonstrate experimentally that SCIGAN outperforms GPS and DRNets.

For a discussion of works that address treatment-response estimation without a dosage parameter, see Appendix A. Note that for such methods we cannot treat the dosage as an input due to the bias associated with its assignment. In Appendix A, we also describe the relationships between our work (causal inference for continuous interventions) and policy optimization with continuous treatments.

3 Problem formulation

We consider receiving observations of the form $(\mathbf{x}^i, t_f^i, y_f^i)$ for $i = 1, ..., N$, where, for each i, these are independent realizations of the random variables $(\mathbf{X}, T_f, Y_f)$. We refer to $\mathbf{X}$ as the feature vector lying in some feature space $\mathcal{X}$, containing pre-treatment covariates (such as age, weight and lab test results). The treatment random variable, T_f, is in fact a pair of values $T_f = (W_f, D_f)$ where $W_f \in \mathcal{W}$ corresponds to the *type* of treatment being administered (e.g. chemotherapy or radiotherapy) which lies in the discrete space of k treatments, $\mathcal{W} = \{w_1, ..., w_k\}$, and D_f corresponds to the *dosage* of the treatment (e.g. number of cycles, amount of chemotherapy, intensity of radiotherapy), which, for a given treatment w lies in the corresponding treatment's dosage space, $\mathcal{D}_w$ (e.g. the interval $[0, 1]$). We define the set of all treatment-dosage pairs to be $\mathcal{T} = \{(w, d) : w \in \mathcal{W}, d \in \mathcal{D}_w\}$.

Following Rubin's potential outcome framework [21], we assume that for all treatment-dosage pairs, (w, d), there is a potential outcome $Y(w, d) \in \mathcal{Y}$ (e.g. 1-year survival probability). The *observed* outcome is then defined to be $Y_f = Y(W_f, D_f)$. We will refer to the unobserved (potential) outcomes as *counterfactuals*.

The goal is to derive *unbiased* estimates of the potential outcomes for a given set of input covariates:

$$\mu(t, \mathbf{x}) = \mathbb{E}[Y(t)|\mathbf{X} = \mathbf{x}] \tag{1}$$

for each $t \in \mathcal{T}$, $\mathbf{x} \in \mathcal{X}$. We refer to $\mu(\cdot)$ as the individualised dose-response function. A table summarising our notation is given in Appendix B. In order to ensure that this quantity is equal to $\mathbb{E}[Y|\mathbf{X} = \mathbf{x}, T = t]$ and that the dose-response function is identifiable from the observational data, we require the following two assumptions.

Assumption 1. *(Unconfoundedness) The treatment assignment, T_f, and potential outcomes, $Y(w, d)$, are conditionally independent given the covariates $\mathbf{X}$, i.e. $\{Y(w, d)|w \in \mathcal{W}, d \in \mathcal{D}_w\} \perp\!\!\!\perp T_f|\mathbf{X}$.*

Assumption 2. *(Overlap) $\forall \mathbf{x} \in \mathcal{X}$ such that $p(\mathbf{x}) > 0$, we have $1 > p(t|\mathbf{x}) > 0$ for each $t \in \mathcal{T}$.*

4 SCIGAN

We propose estimating μ by first training a generator to generate response curves for each sample *within* the training dataset. The learned generator can then be used to train an inference network using standard supervised methods. We build on the idea presented in [6], using a modified GAN framework to generate potential outcomes conditional on the observed features, treatment and factual outcome. Several changes must be made to both the generator and discriminator architectures and learning paradigms in order to produce a model capable of handling the dose-response setting.

4.1 Counterfactual Generator

Our generator, $\mathbf{G} : \mathcal{X} \times \mathcal{T} \times \mathcal{Y} \times \mathcal{Z} \to \mathcal{Y}^{\mathcal{T}}$ takes features, $\mathbf{x} \in \mathcal{X}$, factual outcome, $y_f \in \mathcal{Y}$, received treatment and dosage, $t_f = (w_f, d_f) \in \mathcal{T}$, and some noise, $\mathbf{z} \in \mathcal{Z}$ (typically multivariate uniform or Gaussian), as inputs. The output will be a dose-response curve for each treatment (as shown in Fig. 1), so that the output is a function from $\mathcal{T}$ to $\mathcal{Y}$, i.e. $\mathbf{G}(\mathbf{x}, t_f, y_f, \mathbf{z})(\cdot) : \mathcal{T} \to \mathcal{Y}$. We can then write

$$\hat{y}_{cf}(t) = \mathbf{G}(\mathbf{x}, t_f, y_f, \mathbf{z})(t) \tag{2}$$

as our generated counterfactual outcome for treatment-dosage pair t. We write $\hat{Y}_{cf}(t) = \mathbf{G}(\mathbf{X}, T_f, Y_f, \mathbf{Z})(t)$ (i.e. the random variable induced by $\mathbf{G}$).

While the job of the counterfactual generator is to generate outcomes for the treatment-dosage pairs which were *not* observed, [6] demonstrated that the performance of the counterfactual generator is improved by adding a supervised loss term that regularises its output for the factual treatment (in our case treatment-dosage pair). We define the supervised loss, $\mathcal{L}_S$, to be

$$\mathcal{L}_S(\mathbf{G}) = \mathbb{E}\left[(Y_f - \mathbf{G}(\mathbf{X}, T_f, Y_f, \mathbf{Z})(T_f))^2 \right], \tag{3}$$

where the expectation is taken over $\mathbf{X}, T_f, Y_f$ and $\mathbf{Z}$.

4.2 Counterfactual Discriminator

As noted in Section 1, our discriminator will act on a random set of points from each of the generated dose-response curves. Similar to [6], we define a discriminator, $\mathbf{D}$, that will attempt to pick out the factual treatment-dosage pair from among the (random set of) generated ones. To handle the complexity arising from multiple treatments, we break down our discriminator into two distinct models: a treatment discriminator and a dosage discriminator.

Formally, let $n_w \in \mathbb{Z}^+$ be the number of dosage levels we will compare for treatment $w \in \mathcal{W}$[3]. For each $w \in \mathcal{W}$, let $\tilde{\mathcal{D}}_w = \{D_1^w, ..., D_{n_w}^w\}$ be a random subset[4] of $\mathcal{D}_w$ of size n_w, where for the factual treatment, W_f, $\tilde{\mathcal{D}}_{W_f}$ contains $n_{W_f} - 1$ random elements along with D_f. We define $\tilde{\mathbf{Y}}_w = (D_i^w, \tilde{Y}_i^w)_{i=1}^{n_w} \in (\mathcal{D}_w \times \mathcal{Y})^{n_w}$ to be the vector of dosage-outcome pairs for treatment w where

$$\tilde{Y}_i^w = \begin{cases} Y_f \text{ if } W_f = w \text{ and } D_f = D_i^w \\ \hat{Y}_{cf}(w, D_i^w) \text{ else} \end{cases} \tag{4}$$

and $\tilde{\mathbf{Y}} = (\tilde{\mathbf{Y}}_w)_{w \in \mathcal{W}}$. We will write $d_j^w, \tilde{\mathbf{y}}_w$ and $\tilde{\mathbf{y}}$ to denote realisations of $D_j^w, \tilde{\mathbf{Y}}_w$ and $\tilde{\mathbf{Y}}$.

The treatment discriminator, $\mathbf{D}_{\mathcal{W}} : \mathbf{X} \times \prod_{w \in \mathcal{W}} (\mathcal{D}_w \times \mathcal{Y})^{n_w} \to [0,1]^k$, takes the features, $\mathbf{x}$, and generated potential outcomes, $\tilde{\mathbf{y}}$, and outputs a probability for each treatment, $w_1, ..., w_k$. Writing $\mathbf{D}_{\mathcal{W}}^w$ to denote the output of $\mathbf{D}_{\mathcal{W}}$ corresponding to treatment w, we define the loss, $\mathcal{L}_{\mathcal{W}}$, to be

$$\mathcal{L}_{\mathcal{W}}(\mathbf{D}_{\mathcal{W}}; \mathbf{G}) = -\mathbb{E}\left[\sum_{w \in \mathcal{W}} \mathbb{I}_{\{W_f = w\}} \log \mathbf{D}_{\mathcal{W}}^w(\mathbf{X}, \tilde{\mathbf{Y}}) + \mathbb{I}_{\{W_f \neq w\}} \log(1 - \mathbf{D}_{\mathcal{W}}^w(\mathbf{X}, \tilde{\mathbf{Y}})) \right], \tag{5}$$

where the expectation is taken over $\mathbf{X}, W_f, D_f, \tilde{\mathbf{Y}}$ and $\{\tilde{\mathcal{D}}_w\}_{w \in \mathcal{W}}$.

Then, for each $w \in \mathcal{W}$, the dosage discriminator, $\mathbf{D}_w : \mathcal{X} \times (\mathcal{D}_w \times \mathcal{Y})^{n_w} \to [0,1]^{n_w}$, is a map that takes the features, $\mathbf{x}$, and generated potential outcomes, $\tilde{\mathbf{y}}_w$, corresponding to treatment w and outputs a probability for each dosage level, $d_1^w, ..., d_{n_w}^w$, in a given realisation of $\tilde{\mathcal{D}}_w$. Writing $\mathbf{D}_w^j$ to denote the output of $\mathbf{D}_w$ corresponding to dosage level D_j^w, we define the loss of each dosage discriminator to be

$$\mathcal{L}_d(\mathbf{D}_w; \mathbf{G}) = -\mathbb{E}\left[\mathbb{I}_{\{W_f = w\}} \sum_{j=1}^{n_w} \mathbb{I}_{\{D_f = D_j^w\}} \log \mathbf{D}_w^j(\mathbf{X}, \tilde{\mathbf{Y}}_w) + \mathbb{I}_{\{D_f \neq D_j^w\}} \log(1 - \mathbf{D}_w^j(\mathbf{X}, \tilde{\mathbf{Y}}_w)) \right], \tag{6}$$

where the expectation is taken over $\mathbf{X}, \tilde{\mathcal{D}}_w, \tilde{\mathbf{Y}}_w, W_f$ and D_f. The $\mathbb{I}_{\{W_f = w\}}$ term ensures that only samples for which the factual treatment is w are used to train dosage discriminator $\mathbf{D}_w$ (otherwise there would be no factual dosage for that sample).

[3] In practice we set all n_w to be the same. The default setting is 5 in the experiments.

[4] In practice, when $\mathcal{D}_w = [0,1]$, each D_j^w is sampled independently and uniformly from $[0,1]$. Note that for each training iteration, $\tilde{\mathcal{D}}_w$ is resampled (see Section 1).

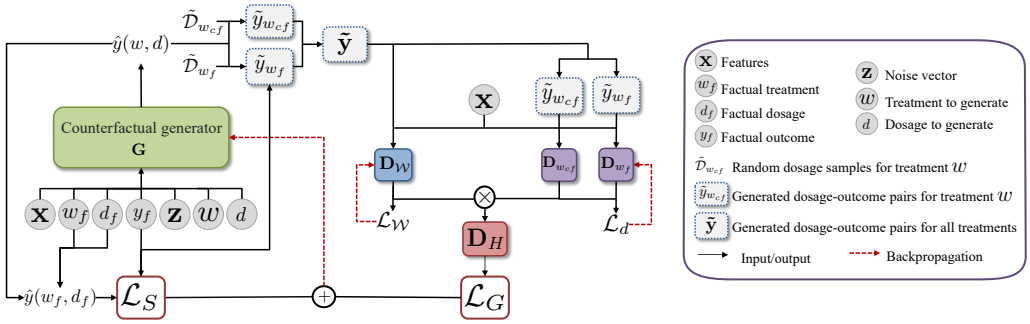


Figure 2: Overview of our model for the setting with two treatments (w_f/w_{cf} being the factual/counterfactual treatment). The generator is used to generate an output for each dosage level in each $\tilde{\mathcal{D}}_w$, these outcomes together with the factual outcome, y_f, are used to create the set of dosage-outcome pairs, $\tilde{\mathbf{y}}$, which is passed to the treatment discriminator. Each dosage discriminator receives only the part of $\tilde{\mathbf{y}}$ corresponding to that treatment, i.e. $\tilde{\mathbf{y}}_w$. These discriminators are combined (Eq. 7) to define $\mathbf{D}_H$ which gives feedback to the generator.

We define the overall discriminator $\mathbf{D}_H : \mathcal{X} \times \prod_{w \in \mathcal{W}} (\mathcal{D}_w \times Y)^{n_w} \to [0,1]^{\sum n_w}$ by defining its output corresponding to the treatment-dosage pair (w, d_j^w) as

$$\mathbf{D}_H^{w,j}(\mathbf{x}, \tilde{\mathbf{y}}) = \mathbf{D}_{\mathcal{W}}^{w}(\mathbf{x}, \tilde{\mathbf{y}}) \times \mathbf{D}_w^j(\mathbf{x}, \tilde{\mathbf{y}}_w)\,. \tag{7}$$

Instead of the standard GAN minimax game, the generator and discriminators are trained according to the minimax game defined by seeking $\mathbf{G}^*$, $\mathbf{D}_H^*$ that solve:

$$\mathbf{G}^* = \arg\min_{\mathbf{G}} \mathcal{L}(\mathbf{D}_H^*; \mathbf{G}) + \lambda \mathcal{L}_S(\mathbf{G}) \qquad \mathbf{D}_H^{*\,w,j} = \mathbf{D}_{\mathcal{W}}^{*\,w} \times \mathbf{D}_w^{*\,j} \tag{8}$$

$$\mathbf{D}_{\mathcal{W}}^* = \arg\min_{\mathbf{D}_{\mathcal{W}}} \mathcal{L}_{\mathcal{W}}(\mathbf{D}_{\mathcal{W}}; \mathbf{G}^*) \qquad \mathbf{D}_w^* = \arg\min_{\mathbf{D}_w} \mathcal{L}_d(\mathbf{D}_w; \mathbf{G}^*), \forall w \in \mathcal{W} \tag{9}$$

Fig. 2 depicts our generator and hierarchical discriminator. Pseudo-code for our algorithm can be found in Appendix D.

4.3 Inference Network

Once we have learned the counterfactual generator, we can use it only to access (generated) dose-response curves for all samples in the dataset. To generate dose-response curves for a new sample we use the counterfactual generator along with the original data to train an inference network, $\mathbf{I} : \mathcal{X} \times \mathcal{T} \to \mathcal{Y}$. Details of the loss and pseudo-code can be found in Appendix E.

4.4 Theoretical Analysis

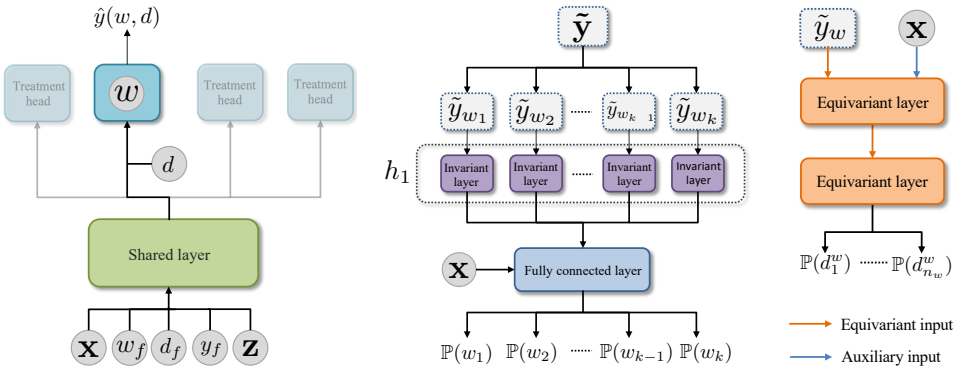
We now state our key theorem: the game defined by Eqs. (8-9) results in our hierarchical GAN learning counterfactuals that agree (in marginal distribution) with the true data.

Theorem 1. *The global minimum of* $\mathcal{L}(\mathbf{D}_H^*; \mathbf{G}) + \lambda \mathcal{L}_S(\mathbf{G})$ *subject to* $\mathbf{D}_H^{*\,w,j} = \mathbf{D}_{\mathcal{W}}^{*\,w} \times \mathbf{D}_w^{*\,j}$, $\mathbf{D}_{\mathcal{W}}^* = \arg\min_{\mathbf{D}_{\mathcal{W}}} \mathcal{L}_{\mathcal{W}}(\mathbf{D}_{\mathcal{W}}; \mathbf{G}^*)$ *and* $\mathbf{D}_w^* = \arg\min_{\mathbf{D}_w} \mathcal{L}_d(\mathbf{D}_w; \mathbf{G}^*), \forall w \in \mathcal{W}$ *is achieved if and only if for all* $\tilde{\mathcal{D}}_w$*, for all* $w, w' \in \mathcal{W}$ *and for all* $d \in \tilde{\mathcal{D}}_w$, $d' \in \tilde{\mathcal{D}}_{w'}$

$$p_{w,d}(\mathbf{y}|\mathbf{x}) = p_{w',d'}(\mathbf{y}|\mathbf{x}) \tag{10}$$

which in turn implies that for any $(w, d) \in \mathcal{T}$ *we have that the generated counterfactual for outcome* (w, d) *for any sample (that was not assigned* (w, d)*) has the same (marginal) distribution (conditional on the features) as the true marginal distribution for that outcome.*

Proof. The proof and intermediate results can be found in Appendix C. □



(a) Generator (b) Treatment Discriminator (c) Dosage Discriminator

Figure 3: Architecture of our generator and discriminators.

5 Architecture

In this section, we describe in detail the novel architectures that we adopt to model each of the functions $\mathbf{G}, \mathbf{D}, \mathbf{D}_{\mathcal{W}}, \mathbf{D}_{w_1}, ..., \mathbf{D}_{w_k}$ which draws from the ideas in [15]. The inference network, $\mathbf{I}$, has the same architecture as the generator, but does not receive w_f, d_f, y_f or $\mathbf{z}$ as inputs.

5.1 Generator Architecture

We adopt a multi-task deep learning model for $\mathbf{G}$ by defining a function $g : \mathcal{X} \times \mathcal{T} \times \mathcal{Y} \times \mathcal{Z} \to \mathcal{H}$ for some latent space $\mathcal{H}$ (typically $\mathbb{R}^l$ for some l) and then for each treatment $w \in \mathcal{W}$ we introduce a multitask "head", $g_w : \mathcal{H} \times \mathcal{D}_w \to \mathcal{Y}$ taking inputs from $\mathcal{H}$ *and* a dosage, d, to produce an outcome $\hat{y}(w,d) \in \mathcal{Y}$. Given observations, $(\mathbf{x}, t_f, y_f)$, a noise vector $\mathbf{z}$, and a target treatment-dosage pair, $t = (w,d)$, we define

$$\mathbf{G}(\mathbf{x}, t_f, y_f, \mathbf{z})(t) = g_w(g(\mathbf{x}, t_f, y_f, \mathbf{z}), d)\,. \tag{11}$$

Each of $g, g_{w_1}, ..., g_{w_k}$ are fully connected networks. A figure of our generator architecture is given in Figure 3(a).

5.2 Hierarchical Discriminator Architectures

In order to ensure the discriminators act as functions of sets, we use ideas from [15] to create permutation invariant and permutation equivariant[5] networks. Zaheer et al. [15] provide several possible building blocks to use to construct invariant and equivariant deep networks. The basic building block we will use for invariant functions will be a layer of the form:

$$f_{inv}(\mathbf{u}) = \sigma(\mathbf{1}_b \mathbf{1}_m^T(\phi(u_1), ..., \phi(u_m)))\,, \tag{12}$$

where $\mathbf{1}_l$ is a vector of 1s of dimension l, ϕ is any function $\phi : \mathcal{U} \to \mathbb{R}^q$ for some q (in this paper we use a standard fully connected layer) and σ is some non-linearity. The basic building block for equivariant functions is defined in terms of equivariance input, $\mathbf{u}$, and auxiliary input, $\mathbf{v}$, by:

$$f_{equi}(\mathbf{u}, \mathbf{v}) = \sigma(\lambda \mathbf{I}_m \mathbf{u} + \gamma(\mathbf{1}_m \mathbf{1}_m^T)\mathbf{u} + (\mathbf{1}_m \Theta^T)\mathbf{v})\,, \tag{13}$$

where $\mathbf{I}_m$ is the $m \times m$ identity matrix, λ and γ are scalar parameters and Θ is a vector of weights.

In the case of the hierarchical discriminator, we want the treatment discriminator, $\mathbf{D}_{\mathcal{W}}$, to be permutation invariant with respect to $\tilde{\mathbf{y}}_w$ for each treatment. To achieve this we define $h_1 : \prod_{w \in \mathcal{W}} (\mathcal{D}_w \times \mathcal{Y})^{n_w} \to \mathcal{H}_H$ and require that h_1 be permutation invariant w.r.t. each of the spaces $(\mathcal{D}_w \times \mathcal{Y})^{n_w}$. We concatenate the output of h_1 with the features $\mathbf{x}$ and pass these through a fully connected network $h_2 : \mathcal{X} \times \mathcal{H}_H \to [0,1]^k$ so that $\mathbf{D}_{\mathcal{W}}(\mathbf{x}, \tilde{\mathbf{y}}) = h_2(\mathbf{x}, h_1(\tilde{\mathbf{y}}))$.

[5] Definitions can be found in [15] and in Appendix F.1.

To construct h_1, we concatenate the outputs of several invariant layers of the form given in Eq. (12) that each individually act on the spaces $(\mathcal{D}_w \times \mathcal{Y})^{n_w}$. That is, for each treatment, $w \in \mathcal{W}$ we define a map $h^w_{inv} : (\mathcal{D}_w \times \mathcal{Y})^{n_w} \to \mathcal{H}^w_H$ by substituting $\tilde{\mathbf{y}}_w$ for $\mathbf{u}$ in Eq. (12). We then define $\mathcal{H}_H = \prod_{w \in \mathcal{W}} \mathcal{H}^w_H$ and $h_1(\tilde{\mathbf{y}}) = (h^{w_1}_{inv}(\tilde{\mathbf{y}}_{w_1}), ..., h^{w_k}_{inv}(\tilde{\mathbf{y}}_{w_k}))$.

We want each dosage discriminator, $\mathbf{D}_w$, to be permutation equivariant with respect to $\tilde{\mathbf{y}}_w$. To achieve this each $\mathbf{D}_w$ will consist of two layers of the form given in Eq. (13) with the equivariance input, $\mathbf{u}$, to the first layer being $\tilde{\mathbf{y}}_w$ and to the second layer being the output of the first layer and the auxiliary input, $\mathbf{v}$, to the first layer being the features, $\mathbf{x}$, and then no auxiliary input to the second layer.

Diagrams depicting the architectures of the treatment discriminator and dosage discriminators can be found in Fig. 3(b) and Fig. 3(c) respectively.

6 Evaluation

The nature of the treatment-effects estimation problem in even the binary treatments setting does not allow for meaningful evaluation on real-world datasets due to the inability to observe the counterfactuals. While there are well-established benchmark synthetic models for use in the binary (or multiple) case, no such models exist for the dosage setting. We propose our own semi-synthetic data simulation to evaluate our model against several benchmarks.

6.1 Experimental setup

Semi-synthetic data generation: We simulate data as follows. We obtain features, $\mathbf{x}$, from a real dataset (in this paper we use TCGA [22], News [19, 23]) and MIMIC III [24])[6]. We consider 3 treatments each accompanied by a dosage. Each treatment, w, is associated with a set of parameters, $\mathbf{v}^w_1, \mathbf{v}^w_2, \mathbf{v}^w_3$. For each run of the experiment, these parameters are sampled randomly by sampling a vector, $\mathbf{u}^w_i$, from $\mathcal{N}(\mathbf{0}, \mathbf{1})$ and then setting $\mathbf{v}^w_i = \mathbf{u}^w_i / ||\mathbf{u}^w_i||$ where $||\cdot||$ is Euclidean norm. The shape of the response curve for each treatment, $f_w(\mathbf{x}, d)$, is given in Table 1, along with a closed-form expression for the optimal dosage. We add $\epsilon \sim \mathcal{N}(0, 0.2)$ noise to the outcomes.

We assign interventions by sampling a dosage, d_w, for each treatment from a beta distribution, $d_w|\mathbf{x} \sim \text{Beta}(\alpha, \beta_w)$. $\alpha \geq 1$ controls the dosage selection bias ($\alpha = 1$ gives the uniform distribution - see Appendix J). $\beta_w = \frac{\alpha - 1}{d^*_w} + 2 - \alpha$, where d^*_w is the optimal dosage[7] for treatment w. This setting of β_w ensures that the mode of $\text{Beta}(\alpha, \beta_w)$ is d^*_w. We then assign a treatment according to $w_f|\mathbf{x} \sim \text{Categorical}(\text{softmax}(\kappa f(\mathbf{x}, d_w))$ where increasing κ increases selection bias, and $\kappa = 0$ leads to random assignments. The factual intervention is given by (w_f, d_{w_f}). Unless otherwise specified, we set $\kappa = 2$ and $\alpha = 2$.

Treatment	Dose-Response	Optimal dosage
1	$f_1(\mathbf{x}, d) = C((\mathbf{v}^1_1)^T\mathbf{x} + 12(\mathbf{v}^1_2)^T\mathbf{x}d - 12(\mathbf{v}^1_3)^T\mathbf{x}d^2)$	$d^*_1 = \frac{(\mathbf{v}^1_2)^T\mathbf{x}}{2(\mathbf{v}^1_3)^T\mathbf{x}}$
2	$f_2(\mathbf{x}, d) = C((\mathbf{v}^2_1)^T\mathbf{x} + \sin(\pi(\frac{\mathbf{v}^{2T}_2\mathbf{x}}{\mathbf{v}^{2T}_3\mathbf{x}})d))$	$d^*_2 = \frac{(\mathbf{v}^2_3)^T\mathbf{x}}{2(\mathbf{v}^2_2)^T\mathbf{x}}$
3	$f_3(\mathbf{x}, d) = C((\mathbf{v}^3_1)^T\mathbf{x} + 12d(d-b)^2$, where $b = 0.75\frac{(\mathbf{v}^3_2)^T\mathbf{x}}{(\mathbf{v}^3_3)^T\mathbf{x}})$	$\frac{b}{3}$ if $b \geq 0.75$ 1 if $b < 0.75$

Table 1: Dose response curves used to generate semi-synthetic outcomes for patient features $\mathbf{x}$. In the experiments, we set $C = 10$. $\mathbf{v}^w_1, \mathbf{v}^w_2, \mathbf{v}^w_3$ are the parameters associated with each treatment w.

Benchmarks: We compare against Generalized Propensity Score (GPS) [16] and Dose Reponse Networks (DRNet) [19] (the standard model and with Wasserstein regularization (DRN-W)). As a baseline, we compare against a standard multilayer perceptron (MLP) that takes patient features, treatment and dosage as input and estimates the patient outcome and a multitask variant (MLP-M) that has a designated head for each treatment. See Appendix K for details of the benchmark models and their hyperparameter optimisation. For metrics, we use Mean Integrated Square Error (MISE), Dosage Policy Error (DPE) and Policy Error (PE) [19, 25]. For further details, see Appendix L.

[6] Details of each dataset can be found in Appendix I

[7] For symmetry, if $d^*_w = 0$, we sample d_w from $1 - \text{Beta}(\alpha, \beta_w)$ where β_w is set as though $d^*_w = 1$.

6.2 Source of gain

Before comparing against the benchmarks, we investigate how each component of our model affects performance. We start with a baseline model in which both the generator and discriminator consist of a single fully connected network. One at a time, we add in the following components (cumulatively until we reach our full model): (1) the supervised loss in Eq. 3 (+ $\mathcal{L}_S$), (2) multitask heads in the generator (+ Multitask), (3) hierarchical discriminator (+ Hierarchical) and (4) invariance/equivariance layers in the treatment and dosage discriminators (+Inv/Eqv). We report the results in Table 2 for TCGA for all 3 error metrics (MISE, DPE and PE), computed over 30 runs (results on News can be found in Appendix M).

	$\sqrt{\text{MISE}}$	$\sqrt{\text{DPE}}$	$\sqrt{\text{PE}}$
Baseline	4.18 ± 0.32	2.06 ± 0.16	1.93 ± 0.12
+ $\mathcal{L}_S$	3.37 ± 0.11	1.14 ± 0.05	0.84 ± 0.05
+ Multitask	3.15 ± 0.12	0.85 ± 0.05	0.67 ± 0.05
+ Hierchical	2.54 ± 0.05	0.36 ± 0.05	0.45 ± 0.05
+ Inv/Eqv	1.89 ± 0.05	0.31 ± 0.05	0.25 ± 0.05

Table 2: Source of gain analysis for our model on the TCGA. Metrics are reported as Mean $\pm$ Std.

The addition of each component results in improved performance, with the final row (our full model) demonstrating the best performance across both datasets and for all metrics. In Appendix M.2 we further compare our hierarchical discriminator with a single network discriminator by investigating both models sensitivity to the hyperparameter n_w. Details of the single discriminator can be found in Appendix G. Architectures for other components of the ablation studies can be found in Appendix H.

6.3 Benchmarks comparison

We now compare SCIGAN[8] against the benchmarks on our 3 semi-synthetic datasets. For MIMIC, due to the low number of samples available, we use two treatments - 2 and 3. We report $\sqrt{\text{MISE}}$ and $\sqrt{\text{PE}}$ in Table 3, $\sqrt{\text{DPE}}$ is given in Appendix M.3. We see that SCIGAN demonstrates a statistically significant improvement over every benchmark across all 3 datasets. In Appendix M.4 we compare SCIGAN with DRNET and GPS for an increasing number of treatments.

Method	**TCGA**		**News**		**MIMIC**	
	$\sqrt{\text{MISE}}$	$\sqrt{\text{PE}}$	$\sqrt{\text{MISE}}$	$\sqrt{\text{PE}}$	$\sqrt{\text{MISE}}$	$\sqrt{\text{PE}}$
SCIGAN	$\mathbf{1.89 \pm 0.05}$	$\mathbf{0.25 \pm 0.05}$	$\mathbf{3.71 \pm 0.05}$	$\mathbf{3.90 \pm 0.05}$	$\mathbf{2.09 \pm 0.12}$	$\mathbf{0.32 \pm 0.05}$
DRNet	3.64 ± 0.12	0.67 ± 0.05	4.98 ± 0.12	4.17 ± 0.11	4.45 ± 0.12	1.44 ± 0.05
DRN-W	3.71 ± 0.12	0.63 ± 0.05	5.07 ± 0.12	4.56 ± 0.12	4.47 ± 0.12	1.37 ± 0.05
GPS	4.83 ± 0.01	1.60 ± 0.01	6.97 ± 0.01	24.1 ± 0.05	7.39 ± 0.00	20.2 ± 0.01
MLP-M	3.96 ± 0.12	1.20 ± 0.05	5.17 ± 0.12	5.82 ± 0.16	4.97 ± 0.16	1.59 ± 0.05
MLP	4.31 ± 0.05	0.97 ± 0.05	5.48 ± 0.16	6.45 ± 0.21	5.34 ± 0.16	1.65 ± 0.05

Table 3: Performance of individualized treatment-dose response estimation on three datasets. Bold indicates the method with the best performance for each dataset. Metrics are reported as Mean $\pm$ Std.

6.4 Discrete dosages

In this experiment, we investigate the discrete dosage setting. Details of the experimental setup can be found in Appendix M.6. We report Mean Squared Error (MSE) of SCIGAN and GANITE in Fig. 4 where we vary the number of discrete dosages from 3 to 30. We see that GANITE is incapable of handling more than 7 discrete dosages, whereas the hierarchical discriminator together with the invariant and equivariant layers allow SCIGAN to maintain performance as the number of dosages increases. Importantly, this demonstrates SCIGAN's wide-ranging applicability in both discrete and continuous settings.

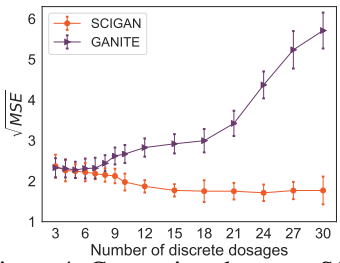
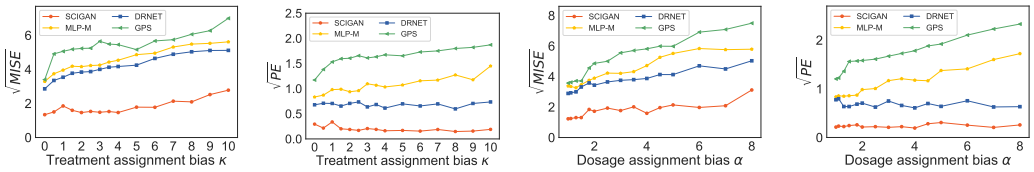


Figure 4: Comparison between SCIGAN and GANITE.

[8]The implementation of SCIGAN can be found at `https://bitbucket.org/mvdschaar/mlforhealthlabpub/` and at `https://github.com/ioanabica/SCIGAN`.

6.5 Treatment and dosage selection bias

Finally, we assess each model's robustness to treatment and dosage bias. We report $\sqrt{\text{MISE}}$ and $\sqrt{\text{PE}}$ on TCGA here. For the other metrics see Appendix M.8. Fig. 5(a) shows the performance of the 4 methods for κ between 0 (no bias) and 10 (strong bias). Fig. 5(b) shows the performance for α between 1 (no bias) and 8 (strong bias). SCIGAN shows consistent performance, significantly outperforming the benchmarks for all κ and α.



(a) Treatment bias (b) Treatment bias (c) Dosage bias (d) Dosage bias

Figure 5: Performance of the 4 methods on datasets with varying bias levels.

7 Conclusion

In this paper we proposed a novel framework for estimating response curves for continuous interventions from observational data. We provided theoretical justification for our use of a modified GAN framework, which introduced a novel hierarchical discriminator. While our approach is very flexible (it accepts multiple treatments potentially with/without dosage parameters), one limitation is that SCIGAN needs at least a few thousand training samples, as is generally the case with neural networks, and GANs in particular. As for future research, we used out-of-the-box methods for the invariant and equivariant layers but feel that more can be done to make these layers as expressive as possible for this model. Moreover, another important research direction is to extend this work to the temporal causal inference setting [26–29] and estimate counterfactual outcomes for sequences of continuous interventions.

Broader Impact

The impact of this problem in the healthcare setting is clear - being able to better estimate individualised responses to dosages will help us select treatments that result in improved patient outcomes. Moreover, clinicians and patients will often need to consider several different outcomes (such as potential side effects); better estimates of such outcomes allow the patients to make a more informed decision that is suitable for them.

Going beyond predictions by using causal inference methods to estimate the effect of interventions will result in more accurate and robust estimates and thus create more reliable components for use as part of decision support systems. Much of the recent work in causal inference has focused on binary or categorical treatments. Nevertheless, continuous interventions arise in many practical scenarios and building reliable methods for estimating their effects is paramount. We believe that our proposed model, SCIGAN, represents an important step forward in this direction. Nevertheless, we acknowledge the fact that the work presented in this paper is on the theoretical side and significant testing, potentially through clinical trials, will be needed before such methods can be used in practice, particularly due to the life-threatening implications of incorrect estimates. The risk of incorrectly assigning treatments can be significantly mitigated by ensuring that such models are used as *support* systems alongside clinicians, rather than instead of clinicians.

All work towards better estimating and understanding interventions can be used negatively, where someone wishing to cause harm can use the estimated outcomes to select the worst outcome.

Acknowledgments

We would like to thank the reviewers for their valuable feedback. The research presented in this paper was supported by The Alan Turing Institute, under the EPSRC grant EP/N510129/1 and by the US Office of Naval Research (ONR), NSF 1722516.

References

[1] Dimitris Bertsimas, Nathan Kallus, Alexander M Weinstein, and Ying Daisy Zhuo. Personalized diabetes management using electronic medical records. *Diabetes Care*, 40(2):210–217, 2017.

[2] Ahmed M Alaa, Michael Weisz, and Mihaela Van Der Schaar. Deep counterfactual networks with propensity-dropout. *ICML 2017 - Workshop on Principled Approaches to Deep Learning*, 2017.

[3] Ahmed M Alaa and Mihaela van der Schaar. Bayesian inference of individualized treatment effects using multi-task gaussian processes. In *Advances in Neural Information Processing Systems*, pages 3424–3432, 2017.

[4] Susan Athey and Guido Imbens. Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360, 2016.

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

[6] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. GANITE: Estimation of individualized treatment effects using generative adversarial nets. *International Conference on Learning Representations (ICLR)*, 2018.

[7] Ahmed Alaa and Mihaela Schaar. Limits of estimating heterogeneous treatment effects: Guidelines for practical algorithm design. In *International Conference on Machine Learning*, pages 129–138, 2018.

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

[9] Ioana Bica, Ahmed M Alaa, Craig Lambert, and Mihaela van der Schaar. From real-world patient data to individualized treatment effects using machine learning: Current and future methods to address underlying challenges. *Clinical Pharmacology & Therapeutics*, 2020.

[10] Donna Döpp-Zemel and AB Johan Groeneveld. High-dose norepinephrine treatment: determinants of mortality and futility in critically ill patients. *American Journal of Critical Care*, 22(1):22–32, 2013.

[11] Kyle Wang, Michael J Eblan, Allison M Deal, Matthew Lipner, Timothy M Zagar, Yue Wang, Panayiotis Mavroidis, Carrie B Lee, Brian C Jensen, Julian G Rosenman, et al. Cardiac toxicity after radiotherapy for stage III non–small-cell lung cancer: Pooled analysis of dose-escalation trials delivering 70 to 90 Gy. *Journal of Clinical Oncology*, 35(13):1387, 2017.

[12] William R Zame, Ioana Bica, Cong Shen, Alicia Curth, Hyun-Suk Lee, Stuart Bailey, James Weatherall, David Wright, Frank Bretz, and Mihaela van der Schaar. Machine learning for clinical trials in the era of covid-19. *Statistics in Biopharmaceutical Research*, pages 1–12, 2020.

[13] Peter Spirtes. A tutorial on causal inference. 2009.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[15] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017.

[16] Guido W Imbens. The role of the propensity score in estimating dose-response functions. *Biometrika*, 87(3):706–710, 2000.

[17] Kosuke Imai and David A Van Dyk. Causal inference with general treatment regimes: Generalizing the propensity score. *Journal of the American Statistical Association*, 99(467):854–866, 2004.

[18] Keisuke Hirano and Guido W Imbens. The propensity score with continuous treatments. *Applied Bayesian Modeling and Causal Inference from Incomplete-Data Perspectives*, 226164:73–84, 2004.

[19] Patrick Schwab, Lorenz Linhardt, Stefan Bauer, Joachim M Buhmann, and Walter Karlen. Learning counterfactual representations for estimating individual dose-response curves. *arXiv preprint arXiv:1902.00981*, 2019.

[20] Uri Shalit, Fredrik D Johansson, and David Sontag. Estimating individual treatment effect: Generalization bounds and algorithms. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3076–3085. JMLR. org, 2017.

[21] Donald B Rubin. Bayesianly justifiable and relevant frequency calculations for the applies statistician. *The Annals of Statistics*, pages 1151–1172, 1984.

[22] John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. The cancer genome atlas pan-cancer analysis project. *Nature Genetics*, 45(10):1113, 2013.

[23] Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *International Conference on Machine Learning*, pages 3020–3029, 2016.

[24] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3:160035, 2016.

[25] Ricardo Silva. Observational-interventional priors for dose-response learning. In *Advances in Neural Information Processing Systems*, pages 1561–1569, 2016.

[26] James M Robins, Miguel Angel Hernan, and Babette Brumback. Marginal structural models and causal inference in epidemiology, 2000.

[27] Bryan Lim, Ahmed Alaa, and Mihaela van der Schaar. Forecasting treatment responses over time using recurrent marginal structural networks. In *Advances in Neural Information Processing Systems*, pages 7493–7503, 2018.

[28] Ioana Bica, Ahmed M Alaa, James Jordon, and Mihaela van der Schaar. Estimating counterfactual treatment outcomes over time through adversarially balanced representations. In *International Conference on Learning Representations*, 2020.

[29] Ioana Bica, Ahmed M Alaa, and Mihaela van der Schaar. Time series deconfounder: Estimating treatment effects over time in the presence of hidden confounders. *International Conference on Machine Learning (ICML)*, 2020.

[30] J Stoehlmacher, DJ Park, W Zhang, D Yang, S Groshen, S Zahedy, and HJ Lenz. A multivariate analysis of genomic polymorphisms: Prediction of clinical outcome to 5-FU/Oxaliplatin combination chemotherapy in refractory colorectal cancer. *British Journal of Cancer*, 91(2):344, 2004.

[31] Min Qian and Susan A Murphy. Performance guarantees for individualized treatment rules. *Annals of Statistics*, 39(2):1180, 2011.

[32] Richard K Crump, V Joseph Hotz, Guido W Imbens, and Oscar A Mitnik. Nonparametric tests for treatment effect heterogeneity. *The Review of Economics and Statistics*, 90(3):389–405, 2008.

[33] Claudia Shi, David M Blei, and Victor Veitch. Adapting neural networks for the estimation of treatment effects. *arXiv preprint arXiv:1906.02120*, 2019.

[34] Hugh A Chipman, Edward I George, Robert E McCulloch, et al. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

[35] Nathan Kallus. Recursive partitioning for personalization using observational data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1789–1798. JMLR. org, 2017.

[36] Sheng Li and Yun Fu. Matching on balanced nonlinear representations for treatment effects estimation. In *Advances in Neural Information Processing Systems*, pages 929–939, 2017.

[37] Liuyi Yao, Sheng Li, Yaliang Li, Mengdi Huai, Jing Gao, and Aidong Zhang. Representation learning for treatment effect estimation from observational data. In *Advances in Neural Information Processing Systems*, pages 2633–2643, 2018.

[38] Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *The Journal of Machine Learning Research*, 16(1):1731–1755, 2015.

[39] Nathan Kallus and Angela Zhou. Policy evaluation and optimization with continuous treatments. *arXiv preprint arXiv:1802.06037*, 2018.

[40] Dimitris Bertsimas and Christopher McCord. Optimization over continuous and multi-dimensional decisions with observational data. In *Advances in Neural Information Processing Systems*, pages 2962–2970, 2018.

[41] Victor Chernozhukov, Mert Demirer, Greg Lewis, and Vasilis Syrgkanis. Semi-parametric efficient policy learning with continuous actions. In *Advances in Neural Information Processing Systems*, pages 15039–15049, 2019.

[42] Douglas Galagate. *Causal Inference with a Continuous Treatment and Outcome: Alternative Estimators for Parametric Dose-Response function with Applications.* PhD thesis, 2016.

[43] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

A Expanded Related Works

Most methods for performing causal inference in the static setting focus on the scenario with two or multiple treatment options and no dosage parameter. The approaches taken by such methods to estimate the treatment effects involve either building a separate regression model for each treatment [1, 30, 31] or using the treatment as a feature and adjusting for the imbalance between the different treatment populations. The former does not generalise to the dosage setting due to the now infinite number of possible treatments available. In the latter case, methods for handling the selection bias involve propensity weighting [2, 32, 33], building sub-populations using tree based methods [4, 5, 34, 35] or building balancing representations between patients receiving the different treatments [20, 23, 36, 37]. An additional approach involves modelling the data distribution of the factual and counterfactual outcomes [3, 6].

[25] leverages observational and interventional data to estimate the effects of discrete dosages for a single treatment. In particular, [25] uses observational data to construct a non-stationary covariance function and develop a hierarchical Gaussian process prior to build a distribution over the dose response curve. Then, controlled interventions are employed to learn a non-parametric affine transform to reshape this distribution. The setting in [25] differs significantly from ours as we do not assume access to any interventional data.

A.1 Comparison with GANITE

Naive attempts to extend [6] to the continuous setting might involve: (1) discretising the continuous space of interventions; (2) somehow passing entire response curves to the discriminator and asking it to identify the point on the curve that corresponds to the factual outcome. Naturally, discretisation comes with a cost. If the discretisation is too coarse, the response curves will not be well-approximated. On the other hand, we show experimentally that GANITE is incapable of handling a high number of discrete interventions (corresponding to having a finer discretisation). In fact, although SCIGAN was designed for continuous interventions, it can be applied in the discrete setting and we show that it outperforms GANITE when the (discrete) parameter space is not small.

For (2), the problem is in defining a mechanism for generating these response curves in a form that can be passed to the discriminator and ensuring the *continuity* of these curves around the factual outcome so that the discontinuity itself does not make identification trivial for the discriminator. To overcome this we define a discriminator that acts on a finite set of points from each generated response curve (rather than on entire curves), as shown in Fig. 1. From *among the chosen points*, the discriminator attempts to identify the factual one. The set of points is sampled randomly *each* time an input would be passed to the discriminator. As our discriminator will be acting on a *set* of random intervention-outcome pairs, we explicitly condition it to behave as a function on a set. In particular, we draw on ideas from [15] to ensure that its output does not depend on the *order* of its input.

In addition, for the setting in which there are multiple possible interventions that *each* have an associated continuous parameter (which is the main setting of the paper), we propose a *hierarchical* discriminator which breaks down the job of the discriminator into determining the factual intervention and determining the factual parameter using separate networks. We show in the experiments section that this approach significantly improves performance and is more stable than using a single network discriminator. In this setting, we also model the generator as a multi-task deep network capable of taking a continuous parameter as an input; this gives us the flexibility to learn heterogeneous response curves for the different interventions.

A.2 Off-policy evaluation and policy optimization with continuous treatments

A problem related to ours involves evaluating policies and learning optimal policies from logged data [38]. In this context, several methods have been proposed for performing off-policy evaluation and policy optimization with continuous treatments [39–41]. It should be emphasized that there is a significant difference between this line of research, which aims to find optimal policies, and the causal inference setting considered in this paper, where the aim is to learn the counterfactual patient outcomes under all possible treatment options.

For off-policy policy evaluation with continuous treatments, [39] propose a method based on inverse-propensity weighting to estimate the value function, i.e. cumulative reward of a target policy from

observational data. However, their proposed method does not perform any type of ITE estimation and does not have any intermediate per-sample value estimates. Alternatively, [41] does perform intermediate estimation of the value function for each sample. However, their contribution is to construct a doubly-robust estimator for the value function which crucially depends on knowing (or having some prior knowledge of) the parametric form of the response curve's dependency on the treatment. Without an assumed form, it is not possible to construct their estimator. Our proposed model, SCIGAN, does not assume any prior knowledge about the response curve.

The problem of individualised treatment effect estimation (which is our primary goal) is harder than off-policy evaluation as it involves estimating the outcomes (which may differ from the value function) for every sample and every possible action that could have been taken for that sample. After having learned these counterfactuals it is possible to perform policy evaluation, but note that the use cases for treatment effect estimation go beyond policy evaluation. In particular, for a given setting it may be far more beneficial to present a patient with their estimated outcome along with potential side effects and allow them (or the clinician) to come to a decision based on these several factors. In such a setting, each patient may have their own "internal" value function that depends on potential outcomes and side effects differently.

B Notation

In the table below, we summarise the notation used in our paper. Note that realisations of random variables are denoted using lowercase and subscripts/superscripts used with vector-valued functions denotes their output at the position of the given subscript/superscript.

$\mathcal{X}$	Feature space
$\mathcal{Y}$	Outcome space
$\mathcal{T}$	Intervention space
$\mathcal{W} = \{w_1, ..., w_k\}$	Set of treatments
$\mathcal{D}_w$	Dosage space for treatment $w \in \mathcal{W}$
$\mathbf{X} \in \mathcal{X}$	Features (random variable)
$Y : \mathcal{T} \to \mathcal{Y}$	Potential outcome function (function-valued random variable)
$T_f = (W_f, D_f) \in \mathcal{T}$	Factual/observed intervention (treatment-dosage pair) (random variable)
$Y_f \in \mathcal{Y}$	Outcome corresponding to the observed intervention ($Y_f = Y(W_f, D_f)$)
$\mathbf{G}$	Generator
$\mathbf{Z}$	Random noise (for input to generator) (random variable)
$\hat{Y}_{cf} : \mathcal{T} \to \mathcal{Y}$	Counterfactual outcome function induced by $\mathbf{G}$
$\mathbf{D}$	Discriminator
$\tilde{\mathcal{D}}_w = \{D_1^w, ..., D_{n_w}^w\}$	Random (finite) subset of $\mathcal{D}_w$
n_w	Size of $\tilde{\mathcal{D}}_w$ (i.e. number of dosage levels passed to discriminator for treatment $w \in \mathcal{W}$)
$\tilde{\mathbf{Y}}_w = (D_i^w, \tilde{Y}_i^w)_{i=1}^{n_w}$	Vector of dosage-outcome pairs generated by $\mathbf{G}$ (and Y_f) using $\tilde{\mathcal{D}}_w$
$\mathbf{D}_{\mathcal{W}}$	Treatment discriminator
$\mathbf{D}_w$	Dosage discriminator for treatment $w \in \mathcal{W}$
$\mathbf{D}_H$	Hierarchical discriminator defined by combining $\mathbf{D}_{\mathcal{W}}$ and $\mathbf{D}_w$
$\mathbf{I}$	Inference network
$\mathcal{L}$	GAN loss
$\mathcal{L}_S$	Supervised loss

C Proofs of Theoretical Results

In this section we prove Theorem 1. Note that these results also apply to GANITE (with very minor modifications - the proofs are even simpler in the case of GANITE).

In order to prove Theorem 1, we analyse the simpler minimax game defined by

$$\min_{\mathbf{G}} \max_{\mathbf{D}} \mathcal{L}(\mathbf{D}, \mathbf{G}) + \lambda \mathcal{L}_S(\mathbf{G}), \tag{14}$$

which corresponds to the single discriminator model (instead of the hierarchical model) and then use this to prove our full result.

Lemma 1. *Fix* $\mathbf{G}$ *and* $\tilde{\mathcal{D}} = \bigcup_w \tilde{\mathcal{D}}_w$. *Let* $p_{w,d}(\mathbf{y}|\mathbf{x}) = p_r(y_{w,d}|\mathbf{x}) p_{\mathbf{G}}(\mathbf{y}_{\neg w,d}|\mathbf{x}, y_{w,d})$ *denote the induced joint density of outcomes when restricted to dosages in* $\tilde{\mathcal{D}}$, *where* p_r *denotes the true density that generated the observed outcome and* $p_{\mathbf{G}}$ *denotes the density induced by* $\mathbf{G}$ *over the remaining dosages in* $\tilde{\mathcal{D}}$. *Then the optimal discriminator is*

$$\mathbf{D}^*_{w,j}(\mathbf{x}, \mathbf{y}) = \frac{\tilde{p}(w, d_j|\mathbf{x}) p_{w,d_j}(\mathbf{y}|\mathbf{x})}{\sum_{w' \in \mathcal{W}} \sum_{i=1}^{n_w} \tilde{p}(w', d_i|\mathbf{x}) p_{w',d_i}(\mathbf{y}|\mathbf{x})} \tag{15}$$

where $\tilde{p}$ *is the* $\tilde{\mathcal{D}}$*-restricted propensity given by* $\tilde{p}(w, d_j|\mathbf{x}) = p(w|\mathbf{x})(p(d_j|\mathbf{x}, w) / \sum_{i=1}^{n_w} p(d_i|\mathbf{x}, w))$.

Proof. Fix $\mathbf{G}$ and $\tilde{\mathcal{D}} = \bigcup_w \tilde{\mathcal{D}}_w$. The optimal discriminator is given by $\arg\min_{\mathbf{D}} \mathcal{L}(\mathbf{D}, \mathbf{G})$. We have

$$\mathcal{L}(\mathbf{D}, \mathbf{G}) = \mathbb{E}\left[\sum_{w \in \mathcal{W}} \sum_{d \in \tilde{\mathcal{D}}_w} \mathbb{I}_{\{T_f = (w,d)\}} \log \mathbf{D}^{w,d}(\mathbf{X}, \tilde{\mathbf{Y}}) + \mathbb{I}_{\{T_f \neq (w,d)\}} \log(1 - \mathbf{D}^{w,d}(\mathbf{X}, \tilde{\mathbf{Y}})) \right] \tag{16}$$

$$= \mathbb{E}_{\tilde{\mathcal{D}}}\Bigg[\sum_{w \in \mathcal{W}} \sum_{d \in \tilde{\mathcal{D}}_w} \int_{(\mathbf{x},\mathbf{y})} \tilde{p}(w, d|\mathbf{x}) p_{w,d}(\mathbf{y}|\mathbf{x}) \log \mathbf{D}^{w,d}(\mathbf{x}, \mathbf{y}) + \Bigg(\sum_{w', d' \neq w, d} \tilde{p}(w', d'|\mathbf{x}) p_{w',d'}(\mathbf{y}|\mathbf{x}) \Bigg) \log(1 - \mathbf{D}^{w,d}(\mathbf{x}, \mathbf{y})) p(\mathbf{x}) d\mathbf{y} d\mathbf{x} \Bigg] \tag{17}$$

where we have taken the (conditional on $\tilde{\mathcal{D}}$) expectations inside the sums and replaced indicator functions with densities as appropriate. We now note that $a \log p + b \log(1-p)$ for $p \in (0,1)$ has a unique maximum at $p = \frac{a}{a+b}$, thus implying that the integrand is maximised when

$$\mathbf{D}^{w,d}(\mathbf{x}, \mathbf{y}) = \frac{\tilde{p}(w, d|\mathbf{x}) p_{w,d}(\mathbf{y}|\mathbf{x})}{\tilde{p}(w, d|\mathbf{x}) p_{w,d}(\mathbf{y}|\mathbf{x}) + \sum_{w', d' \neq w, d} \tilde{p}(w', d'|\mathbf{x}) p_{w',d'}(\mathbf{y}|\mathbf{x})}. \tag{18}$$

This gives the required result. □

Using Lemma 1 we can now show that the optimal solution to our single discriminator game is when the marginal distributions of the generated counterfactuals are equal to the true counterfactuals. Importantly, this suffices for estimating μ since the expectation is only concerned with the marginal distribution of $Y(w, d)$.

Lemma 2. *The global minimum of the minimax game defined by* $\min_{\mathbf{G}} \max_{\mathbf{D}} \mathcal{L}(\mathbf{D}, \mathbf{G})$ *is achieved if and only if for all* $\tilde{\mathcal{D}}_w$, *for all* $w, w' \in \mathcal{W}$ *and for all* $d \in \tilde{\mathcal{D}}$, $d' \in \tilde{\mathcal{D}}_{w'}$ *we have that*

$$p_{w,d}(\mathbf{y}|\mathbf{x}) = p_{w',d'}(\mathbf{y}|\mathbf{x}) \tag{19}$$

which in turn implies that for any treatment-dosage pair $(w, d) \in \mathcal{T}$ *we have that the generated counterfactual for outcome* (w, d) *for any sample (that was not assigned* (w, d)*) has the same (marginal) distribution (conditional on the features) as the true marginal distribution for that outcome.*

Proof. For fixed $\tilde{\mathcal{D}}$ and $\mathbf{x}$ we note that by substituting the optimal discriminator into $\mathcal{L}(\mathbf{D}, \mathbf{G})$ and subtracting $\sum_{w \in \mathcal{W}} \sum_{i=1}^{n_w} \log \tilde{p}(w, d_i|\mathbf{x})$ (which is independent of $\mathbf{G}$) we obtain

$$\mathcal{L}(\mathbf{D}^*, \mathbf{G}) - \int_{\mathbf{x}} \left(\sum_{w \in \mathcal{W}} \sum_{i=1}^{n_w} \log \tilde{p}(w, d_i|\mathbf{x}) \right) p(\mathbf{x}) d\mathbf{x} \tag{20}$$

$$= \mathbb{E}_{\tilde{\mathcal{D}}} \int_{\mathbf{x}} \mathrm{KL}\Big(p_{w,d}(\mathbf{y}|\mathbf{x}) || \hat{p}(\mathbf{y}|\mathbf{x}) \Big) + \mathrm{KL}\Big(\frac{1}{1 - \tilde{p}(w, d|\mathbf{x})} \sum_{t' \neq (w,d)} \tilde{p}(t'|\mathbf{x}) p_{t'}(\mathbf{y}|\mathbf{x}) || \hat{p}(\mathbf{y}|\mathbf{x}) \Big) d\mathbf{x} \tag{21}$$

where KL is the KL divergence and $\hat{p}(\mathbf{y}|\mathbf{x}) = \sum_{t \in \tilde{\mathcal{T}}} \tilde{p}(t|\mathbf{x}) p_t(\mathbf{y}|\mathbf{x})$ where $\tilde{\mathcal{T}}$ is the restriction of $\mathcal{T}$ to the dosages in $\tilde{\mathcal{D}}$. We then note that the KL divergence is minimised if and only if the two densities are equal, and we note by definition of $\hat{p}$ this occurs if and only if $p_{w,d}(\mathbf{y}|\mathbf{x}) = p_{w',d'}(\mathbf{y}|\mathbf{x})$ for all w, d, w', d'. This also directly implies that the marginal distributions for any fixed treatment-dosage pair agree for all factually observed treatments. In particular, if a sample received treatment $t' \neq t$, we have that the counterfactual generated for t for this sample has the same distribution as the true data generating distribution. □

Finally, we prove the following result, from which Theorem 1 follows immediately.

Theorem 1. *An optimal solution to the game defined by Equations 8 - 9 is also an optimal solution to the game defined by Equation 14 if the response curves generated by the generator for different treatments are conditionally independent given the features.*

Proof. To prove this result, it suffices to show that for fixed $\mathbf{G}$, $\mathbf{D}_H^* = \mathbf{D}^*$. To show this, we observe that by the same arguments as given for Lemma 1, we have the following:

$$\mathbf{D}_{\mathcal{W}}^{w\,*}(\mathbf{x}, \mathbf{y}) = \frac{p(w|\mathbf{x}) \Big(\sum_{i=1}^{n_w} \tilde{p}(d_i|\mathbf{x}, w) p_{w,d_i}(\mathbf{y}|\mathbf{x}) \Big)}{\sum_{w' \in \mathcal{W}} \Big(p(w'|\mathbf{x}) \sum_{i=1}^{n_w} \tilde{p}(d_i|\mathbf{x}, w) \Big)} \tag{22}$$

$$\mathbf{D}_w^{j\,*}(\mathbf{x}, \mathbf{y}_w) = \frac{\tilde{p}(d_j|\mathbf{x}, w) p_{w,d_j}(\mathbf{y}_w|\mathbf{x})}{\sum_{i=1}^{n_w} \tilde{p}(d_i|\mathbf{x}, w) p_{w,d_i}(\mathbf{y}_w|\mathbf{x})} \tag{23}$$

where $\mathbf{y}_w$ is the restriction of $\mathbf{y}$ to the outcomes corresponding to treatment w. By multiplying (23) by $\frac{p_{w,d_j}(\mathbf{y}_{\neq w}|\mathbf{y},\mathbf{x})}{p_{w,d_j}(\mathbf{y}_{\neq w}|\mathbf{y},\mathbf{x})}$ we obtain

$$\mathbf{D}_w^{j\,*}(\mathbf{x}, \mathbf{y}_w) = \frac{\tilde{p}(d_j|\mathbf{x}, w) p_{w,d_j}(\mathbf{y}|\mathbf{x})}{\sum_{i=1}^{n_w} \tilde{p}(d_i|\mathbf{x}, w) p_{w,d_i}(\mathbf{y}|\mathbf{x})} \tag{24}$$

since the conditional independence assumption implies that $p_{w,d_j}(\mathbf{y}_{\neq w}|\mathbf{y}, \mathbf{x}) = p_{w,d_i}(\mathbf{y}_{\neq w}|\mathbf{y}, \mathbf{x})$ for all $i, j = 1, ..., n_w$. Multiplying (22) and (24) together to get $\mathbf{D}_H^{w,j}$, we notice that the denominator in (24) cancels with the bracketed term of the numerator in (22) to give

$$\mathbf{D}_H^{*\,w,j} = \frac{p(w|\mathbf{x}) \tilde{p}(d_j|\mathbf{x}, w) p_{w,d_j}(\mathbf{y}|\mathbf{x})}{\sum_{w' \in \mathcal{W}} \Big(p(w'|\mathbf{x}) \sum_{i=1}^{n_w} \tilde{p}(d_i|\mathbf{x}, w) \Big)} \tag{25}$$

$$= \frac{\tilde{p}(w, d_j|\mathbf{x}) p_{w,d_j}(\mathbf{y}|\mathbf{x})}{\sum_{w' \in \mathcal{W}} \sum_{i=1}^{n_w} \tilde{p}(w', d_i|\mathbf{x}) p_{w',d_i}(\mathbf{y}|\mathbf{x})} \tag{26}$$

which is equal to the optimal discriminator for the single loss given in Lemma 1. □

D Counterfactual Generator Pseudo-code

Algorithm 1 Training of the generator in SCIGAN

Input: dataset $\mathcal{C} = \{(\mathbf{x}^i, t_f^i, y_f^i) : i = 1, ..., N\}$, batch size n_{mb}, number of dosages per treatment n_d, number of discriminator updates per iteration n_D, number of generator updates per iteration n_G, dimensionality of noise n_z, learning rate α
Initialize: θ_G, $\theta_{\mathcal{W}}$, $\{\theta_w\}_{w \in \mathcal{W}}$
while G has not converged **do**
Discriminator updates
  for $i = 1, ..., n_D$ **do**
    Sample $(\mathbf{x}_1, (w_1, d_1), y_1), ..., (\mathbf{x}_{n_{mb}}, (w_{n_{mb}}, d_{n_{mb}}), y_{n_{mb}})$ from $\mathcal{C}$
    Sample generator noise $\mathbf{z}_j = (z_1^j, ..., z_{n_z}^j)$ from Unif$([0,1]^{n_z})$ for $j = 1, ..., n_{mb}$
    for $w \in \mathcal{W}$ **do**
      for $j = 1, ..., n_{mb}$ **do**
        Sample $\tilde{D}_w^j = (d_1^{w,j}, ..., d_{n_d}^{w,j})$ independently and uniformly from $(\mathcal{D}_w)^{n_d}$
        Set $\tilde{\mathbf{y}}_w^j$ according to Eq. 4
      Calculate gradient of dosage discriminator loss

$$g_w \leftarrow \nabla_{\theta_w} - \left[\sum_{\{j: w_j = w\}} \sum_{k=1}^{n_d} \mathbb{I}_{\{d_j = d_k^{w,j}\}} \log \mathbf{D}_w(\mathbf{x}_j, \tilde{\mathbf{y}}_w^j) + \mathbb{I}_{\{d_j \neq d_k^{w,j}\}} \log(1 - \mathbf{D}_w(\mathbf{x}_j, \tilde{\mathbf{y}}_w^j)) \right]$$

      Update dosage discriminator parameters $\theta_w \leftarrow \theta_w + \alpha g_w$
    Set $\tilde{\mathbf{y}}_j = (\tilde{\mathbf{y}}_w^j)_{w \in \mathcal{W}}$
    Calculate gradient of treatment discriminator loss

$$g_{\mathcal{W}} \leftarrow \nabla_{\theta_{\mathcal{W}}} - \left[\sum_{j=1}^{n_{mb}} \sum_{w \in \mathcal{W}} \mathbb{I}_{\{w_j = w\}} \log \mathbf{D}_{\mathcal{W}}(\mathbf{x}_j, \tilde{\mathbf{y}}_j) + \mathbb{I}_{\{w_j \neq w\}} \log(1 - \mathbf{D}_{\mathcal{W}}(\mathbf{x}_j, \tilde{\mathbf{y}}_j)) \right]$$

    Update treatment discriminator parameters $\theta_{\mathcal{W}} \leftarrow \theta_{\mathcal{W}} + \alpha g_{\mathcal{W}}$
  Generator updates
  for $i = 1, ..., n_G$ **do**
    Sample $(\mathbf{x}_1, (w_1, d_1), y_1), ..., (\mathbf{x}_{n_{mb}}, (w_{n_{mb}}, d_{n_{mb}}), y_{n_{mb}})$ from $\mathcal{C}$
    Sample generator noise $\mathbf{z}_j = (z_1^j, ..., z_{n_z}^j)$ from Unif$([0,1]^{n_z})$ for $j = 1, ..., n_{mb}$
    Sample $(\tilde{D}_w^j)_{w \in \mathcal{W}}$ from $\Pi_{w \in \mathcal{W}}(\mathcal{D}_w)^{n_d}$ for $j = 1, ..., n_{mb}$
    Set $\tilde{\mathbf{y}}$ according to Eq. 4
    Calculate gradient of generator loss

$$g_G \leftarrow \nabla_{\theta_G} \left[\sum_{j=1}^{n_{mb}} \sum_{w \in \mathcal{W}} \sum_{l=1}^{n_d} \mathbb{I}_{\{w_j = w, d_j = d_l^{w,j}\}} \log(\mathbf{D}_{\mathcal{W}}^w(\mathbf{x}_j, \tilde{\mathbf{y}}_j)_w \times \mathbf{D}_w^l(\mathbf{x}_j, \tilde{\mathbf{y}}_w^j)_l) \right.$$
$$\left. + \mathbb{I}_{\{w_j \neq w, d_j \neq d_l^{w,j}\}} \log(1 - (\mathbf{D}_{\mathcal{W}}^w(\mathbf{x}_j, \tilde{\mathbf{y}}_j) \times D_w^l(\mathbf{x}_j, \tilde{\mathbf{y}}_w^j))) \right]$$

    Update generator parameters $\theta_G \leftarrow \theta_G + \alpha g_G$
Output: G

E Inference Network

To generate dose-response curves for new samples, we learn an inference network, $\mathbf{I} : \mathbf{X} \times \mathcal{T} \to \mathcal{Y}$. This inference network is trained using the original dataset and the learned counterfactual generator. As with the training of the generator and discriminator, we train using a random set of dosages, $\tilde{\mathcal{D}}_w$. The loss is given by

$$\mathcal{L}_I(\mathbf{I}) = \mathbb{E}\left[\sum_{w \in \mathcal{W}} \sum_{d \in \tilde{\mathcal{D}}_w} (\tilde{Y}(w,d) - \mathbf{I}(\mathbf{X},(w,d)))^2\right], \tag{27}$$

where $\tilde{Y}(w,d)$ is Y_f if $T_f = (w,d)$ or given by the generator if $T_f \neq (w,d)$. The expectation is taken over $\mathbf{X}, T_f, Y_f, \mathbf{Z}$ and $\tilde{\mathcal{D}}_w$.

E.1 Pseudo-code for training the Inference Network

Algorithm 2 Training of the inference network in SCIGAN

Input: dataset $\mathcal{C} = \{(\mathbf{x}^i, t_f^i, y_f^i) : i = 1, ..., N\}$, trained generator $\mathbf{G}$, batch size n_{mb}, number of dosages per treatment n_d, dimensionality of noise n_z, learning rate α
Initialize: θ_I
while $\mathbf{I}$ has not converged **do**
  Sample $(\mathbf{x}_1, (w_1, d_1), y_1), ..., (\mathbf{x}_{n_{mb}}, (w_{n_{mb}}, d_{n_{mb}}), y_{n_{mb}})$ from $\mathcal{C}$
  Sample generator noise $\mathbf{z}_j = (z_1^j, ..., z_{n_z}^j)$ from Unif$([0,1]^{n_z})$ for $j = 1, ..., n_{mb}$
  for $j = 1, ..., n_{mb}$ **do**
    for $w \in \mathcal{W}$ **do**
      Sample $\tilde{D}_w^j = (d_1^{w,j}, ..., d_{n_d}^{w,j})$ independently and uniformly from $(\mathcal{D}_w)^{n_d}$
      Set $\tilde{\mathbf{y}}_w^j$ according to Eq. 4 4
  Calculate gradient of inference network loss

$$g_I \leftarrow -\nabla_{\theta_I}\left[\sum_{j=1}^{n_{mb}} \sum_{w \in \mathcal{W}} \sum_{l=1}^{n_d} (\tilde{\mathbf{y}}_w^j)_l - \mathbf{I}(\mathbf{x}_j, (w, d_l^{w,j}))^2\right]$$

  Update inference network parameters $\theta_I \leftarrow \theta_I + \alpha g_I$
Output: $\mathbf{I}$

F Architecture

F.1 Definitions of Permutation Invariance and Permutation Equivariance

The notions of what it means for a function to be *permutation invariant* and *permutation equivariant* with respect to (a subset of) its inputs are given below in definitions 1 and 2, respectively. Let $\mathcal{U}$, $\mathcal{V}$, $\mathcal{C}$ be some spaces. Let $m \in \mathbb{Z}^+$.

Definition 1. *A function $f : \mathcal{U}^m \times \mathcal{V} \to \mathcal{C}$ is permutation* invariant *with respect to the space $\mathcal{U}^m$ if for every $\mathbf{u} = (u_1, ..., u_m) \in \mathcal{U}^m$, every $v \in \mathcal{V}$ and every permutation, σ, of $\{1, ..., m\}$ we have*

$$f(u_1, ..., u_m, v) = f(u_{\sigma(1)}, ..., u_{\sigma(m)}, v)\,. \tag{28}$$

Definition 2. *A function $f : \mathcal{U}^m \times \mathcal{V} \to \mathcal{C}^m$ is permutation* equivariant *with respect to the space $\mathcal{U}^m$ if for every $\mathbf{u} \in \mathcal{U}^m$, every $v \in \mathcal{V}$ and every permutation, σ, of $\{1, ..., m\}$ we have $f(u_{\sigma(1)}, ..., u_{\sigma(m)}, v) = (f_{\sigma(1)}(\mathbf{u}, v), ..., f_{\sigma(m)}(\mathbf{u}, v))$, where $f_j(\mathbf{u}, v)$ is the jth element of $f(\mathbf{u}, v)$.*

To build up functions that are permutation invariant and permutation equivariant we make the following observations: (1) the composition of any function with a permutation invariant function is permutation invariant, (2) the composition of two permutation equivariant functions is permutation equivariant.

As noted in Section 5.2, the basic building block we use for equivariant functions is defined in terms of equivariance input, $\mathbf{u}$, and auxiliary input, $\mathbf{v}$ by

$$f_{equi}(\mathbf{u}, \mathbf{v}) = \sigma(\lambda \mathbf{I}_m \mathbf{u} + \gamma(\mathbf{1}_m \mathbf{1}_m^T)\mathbf{u} + (\mathbf{1}_m \Theta^T)\mathbf{v})\,. \tag{29}$$

G Single Discriminator Model

In the paper we developed a hierarchical discriminator and demonstrated that it performs significantly better than the single discriminator setup that we now describe in this section.

G.1 Single Discriminator

In the single model, we will aim to learn a single discriminator, $\mathbf{D}$, that outputs $\mathbb{P}((W_f, D_f) = (w, d)|\mathbf{X}, \tilde{\mathcal{D}}_w, \tilde{\mathbf{Y}})$ for each $w \in \mathcal{W}$ and $d \in \tilde{\mathcal{D}}_w$. We will write $\mathbf{D}^{w,d}(\cdot)$ to denote the output of $\mathbf{D}$ that corresponds to the treatment-dosage pair (w, d). We define the loss, $\mathcal{L}_D$, to be

$$\mathcal{L}_D(\mathbf{D}; \mathbf{G}) = -\mathbb{E}\left[\sum_{w\in\mathcal{W}} \sum_{d\in\tilde{\mathcal{D}}_w} \mathbb{I}_{\{T_f=(w,d)\}} \log \mathbf{D}^{w,d}(\mathbf{X}, \tilde{\mathbf{Y}}) + \mathbb{I}_{\{T_f\neq(w,d)\}} \log(1 - \mathbf{D}^{w,d}(\mathbf{X}, \tilde{\mathbf{Y}}))\right] \tag{30}$$

where the expectation is taken over $\mathbf{X}, \{\tilde{\mathcal{D}}_w\}_{w\in\mathcal{W}}, \tilde{\mathbf{Y}}, W_f$ and D_f and we note that the dependence on $\mathbf{G}$ is through $\tilde{\mathbf{Y}}$. Our single discriminator will be trained to minimise this loss directly. The generator GAN-loss, $\mathcal{L}_G$, is then defined by

$$\mathcal{L}_G(\mathbf{G}) = -\mathcal{L}_D(\mathbf{D}^*; \mathbf{G}) \tag{31}$$

where $\mathbf{D}^*$ is the optimal discriminator given by minimising $\mathcal{L}_D$. The generator will be trained to minimise $\mathcal{L}_G + \lambda\mathcal{L}_S$.

G.2 Single Discriminator Architecture

In the case of the single discriminator, we want the output of $\mathbf{D}$ corresponding to each treatment $w \in \mathcal{W}$, i.e. $(\mathbf{D}^{w,1}, ..., \mathbf{D}^{w,n_w})$, to be permutation equivariant with respect to $\tilde{\mathbf{y}}_w$ and permutation invariant with respect to each $\tilde{\mathbf{y}}_v$ for $v \in \mathcal{W}\backslash\{w\}$. To achieve this, we first define a function $f : \prod_{w\in\mathcal{W}}(\mathcal{D}_w \times \mathcal{Y})^{n_w} \to \mathcal{H}_S$ and require that this function be permutation invariant with respect to each of the spaces $(\mathcal{D}_w \times \mathcal{Y})^{n_w}$. For each treatment, $w \in \mathcal{W}$, we introduce a multi-task head, $f_w : \mathcal{X} \times \mathcal{H}_S \times (\mathcal{D}_w \times \mathcal{Y})^{n_w} \to [0, 1]^{n_w}$, and require that each of these functions be permutation equivariant with respect to their corresponding input space $(\mathcal{D}_w \times \mathcal{Y})^{n_w}$ but they can depend on the features, $\mathbf{x} \in \mathcal{X}$, and invariant latent representation coming from f arbitrarily. Writing f_w^j to denote the jth output of f_w, the output of the discriminator given input features, $\mathbf{x}$, and generated outcomes, $\tilde{\mathbf{y}}$, is defined by

$$\mathbf{D}^{w,j}(\mathbf{x}, \tilde{\mathbf{y}}) = f_w^i(\mathbf{x}, f(\tilde{\mathbf{y}}), \tilde{\mathbf{y}}_w). \tag{32}$$

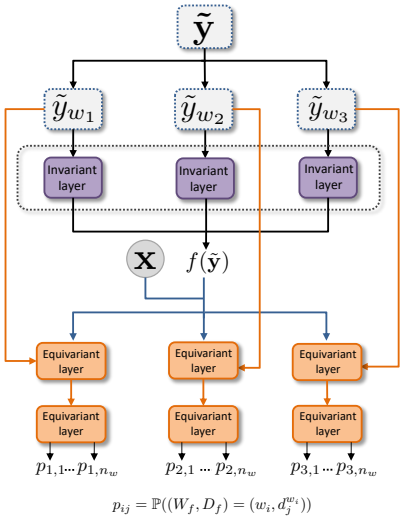


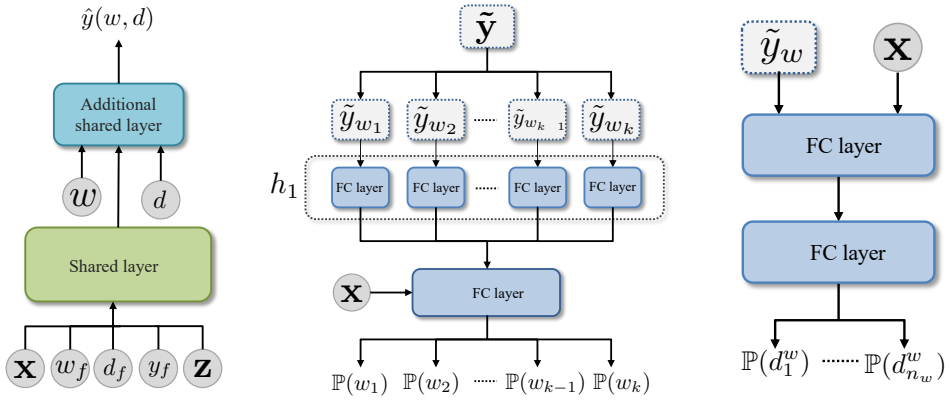
Figure 6: Overview of the single discriminator architecture.

To construct the function f, we concatenate the outputs of several invariant layers of the form given in Eq. 29 that each individually act on the spaces $(\mathcal{D}_w \times \mathcal{Y})^{n_w}$. That is, for each treatment, $w \in \mathcal{W}$ we define a map $f_{inv}^w : (\mathcal{D}_w \times \mathcal{Y})^{n_w} \to \mathcal{H}_S^w$ by substituting $\tilde{\mathbf{y}}_w$ for $\mathbf{u}$ in Eq. 29. We then define $\mathcal{H}_S = \prod_{w\in\mathcal{W}} \mathcal{H}_S^w$ and $f(\tilde{\mathbf{y}}) = (f_{inv}^{w_1}(\tilde{\mathbf{y}}_{w_1}), ..., f_{inv}^{w_k}(\tilde{\mathbf{y}}_{w_k}))$.

Each f_w will consist of two layers of the form given in Eq. (29) with the equivariance input, $\mathbf{u}$, to first layer being $\tilde{\mathbf{y}}_w$ and to the second layer being the output of the first layer and the auxiliary input, $\mathbf{v}$, to the first layer being the concatenation of the features and invariant representation, i.e. $(\mathbf{x}, f(\tilde{\mathbf{y}}))$ and then no auxiliary input to the second layer.

A diagram depicting the architecture of the single discriminator model can be found in Fig. 6.

H Ablation Studies architectures



(a) Generator without multitask heads

(b) Treatment discriminator with fully connected (FC) layers instead of invariant layers

(c) Dosage discriminator with FC layers instead of equivariant layers

Figure 7: Architecture of the generator without multitask and discriminator without the invariant/equivariant layers used in the ablation studies.

I Dataset descriptions

TCGA: The TCGA dataset consists of gene expression measurements for cancer patients [22]. There are 9659 samples for which we used the measurements from the 4000 most variable genes. The gene expression data was log-normalized and each feature was scaled in the $[0, 1]$ interval. For each patient, the features were scaled to have norm 1. We give meaning to our treatments and dosages by considering the treatment as being chemotherapy/radiotherapy/immunotherapy and their corresponding dosages. The outcome can be thought of as the risk of cancer recurrence [19]. We used the same version of the TCGA dataset as used by DRNet `https://github.com/d909b/drnet`.

News: The News dataset consists of word counts for news items. We extracted 10000 news items (randomly sampled) each with 2858 features. As in [19, 23], we give meaning to our treatments and dosages by considering the treatment as being the viewing device (e.g. phone, tablet etc.) used to read the article and the dosage as being the amount of time spent reading it. The outcome can be thought of as user satisfaction. We used the same version of the News dataset as used by DRNet `https://github.com/d909b/drnet`.

MIMIC III: The Medical Information Mart for Intensive Care (MIMIC III) [24] database consists of observational data from patients in the ICU. We extracted 3000 patients (randomly sampled) that receive antibiotics treatment and we used as features 9 clinical covariates, namely age, temperature, heart rate, systolic and diastolic blood pressure, SpO2, FiO2, glucose, and white blood cell count, measured at start of ICU stay. Again, the features were scaled in the $[0, 1]$ interval. In this setting, we can considered as treatments the different antibiotics and their corresponding dosages.

For a summary description of the datasets, see table 4. The datasets are split into 64/16/20% for training, validation and testing respectively. The validation dataset is used for hyperparameter optimization.

	TCGA	**News**	**MIMIC**
Number of samples	9659	10000	3000
Number of features	4000	2858	9
Number of treatments	3*	3	2

Table 4: Summary description of datasets. *: for our final experiment in Appendix M.4 we increase the number of treatments in TCGA to 6 and 9.

J Dosage bias

In order to create dosage-assignment bias in our datasets, we assign dosages according to $d_w|\mathbf{x} \sim \text{Beta}(\alpha, \beta_w)$. The selection bias is controlled by the parameter $\alpha \geq 1$. When we set $\beta_w = \frac{\alpha-1}{d_w^*}+2-\alpha$ (which ensures that the mode of our distribution is d_w^*), we can write the variance of d_w in terms of α and d_w^* as follows

$$\text{Var}(d_w) = \frac{\frac{\alpha^2-\alpha}{d_w^*} + 2\alpha - \alpha^2}{(\frac{\alpha-1}{d_w^*} + 2)^2(\frac{\alpha-1}{d_w^*} + 3)} \approx \frac{c\alpha^2}{d\alpha^3} \,. \tag{33}$$

We see that the variance of our Beta distribution therefore decreases with α, resulting in the sampled dosages being closer to the optimal dosage, thus resulting in higher dosage-selection bias. In addition we note that the $\text{Beta}(1, 1)$ distribution is in fact the uniform distribution, corresponding to the dosages being sampled independently of the patient features, resulting in no selection bias when $\alpha = 1$.

K Benchmarks

We use the publicly available GitHub implementation of DRNet provided by [19]: `https://github.com/d909b/drnet`. Moreover, we also used a GPS implementation similar to the one from `https://github.com/d909b/drnet` which uses the `causaldrf` R package [42]. More spcifically, the GPS implementation uses a normal treatment model, a linear treatment formula and a 2-nd degree polynomial for the outcome. Moreover, for the TCGA and News datasets, we performed PCA and only used the 50 principal components as input to the GPS model to reduce computational complexity.

Hyperparameter optimization: The validation split of the dataset is used for hyperparameter optimization. For the DRNet benchmarks we use the same hyperparameter optimization proposed by [19] with the hyperparameter search ranges described in Table 5. For SCIGAN, we use the hyperparameter optimization method proposed in GANITE [6], where we use the complete dataset from the counterfactual generator to evaluate the MISE on the inference network. We perform a random search [43] for hyperparameter optimization over the search ranges in Table 6. For all experiments with SCIGAN, we used 5000 training iterations for the GAN network and 10000 training iterations for the inference network. This number of training iterations was chosen to ensure convergence of the generator loss, discriminator loss, as well as of the supervised loss. For a fair comparison, for the MLP-M model we used the same architecture used in the inference network of SCIGAN. Similarly, for the MLP model we use the same architecture as for the MLP-M, but without the multitask heads.

Hyperparameter	Search range
Batch size	32, 64, 128
Number of units per hidden layer	24, 48, 96, 192
Number of hidden layers	2, 3
Dropout percentage	0.0, 0.2
Imbalance penalty weight*	0.1, 1.0, 10.0
	Fixed
Number of dosage strata E	5

Table 5: Hyperparameters search range for DRNet. *: For the DRNet model using Wasserstein regularization only.

Hyperparameter	Search range
Batch size	64, 128, 256
Number of units per hidden layer	32, 64, 128
Size of invariant and equivariant representations	16, 32, 64, 128
	Fixed
Number of hidden layers per multitask head	2
Number of dosage samples	5
λ	1
Optimization	Adam Moment Optimization

Table 6: Hyperparameters search range for SCIGAN.

The hyperparameters used to generate the results for SCIGAN are given in Table 7.

The experiments were run on a system with 6CPUs, an Nvidia K80 Tesla GPU and 56GB of RAM.

Hyperparameter	TCGA	News	MIMIC
Batch size	128	256	128
Number of units per hidden layer	64	128	32
Size of invariant and equivariant representations	16	32	16
Number of hidden layers per multitask head	2	2	2
Number of dosage samples	5	5	5
λ	1	1	1

Table 7: Hyperparameters used for obtaining results.

L Metrics

The Mean Integrated Square Error (MISE) measures how well the models estimates the patient outcome across the entire dosage space:

$$\text{MISE} = \frac{1}{N}\frac{1}{k}\sum_{w\in\mathcal{W}}\sum_{i=1}^{N}\int_{\mathcal{D}_w}\left(y^i(w,u) - \hat{y}^i(w,u)\right)^2 \mathrm{d}u\,. \tag{34}$$

In addition to this, we also compute the mean dosage policy error (DPE) [19] to assess the ability of the model to estimate the optimal dosage point for every treatment for each individual:

$$\text{DPE} = \frac{1}{N}\frac{1}{k}\sum_{w\in\mathcal{W}}\sum_{i=1}^{N}\left(y^i(w,d_w^*) - y^i(w,\hat{d}_w^*)\right)^2, \tag{35}$$

where d_w^* is the true optimal dosage and $\hat{d}_w^*$ is the optimal dosage identified by the model. The optimal dosage points for a model are computed using SciPy's implementation of Sequential Least SQuares Programming.

Finally, we compute the mean policy error (PE) [19] which compares the outcome of the true optimal treatment-dosage pair to the outcome of the optimal treatment-dosage pair as selected by the model:

$$\text{PE} = \frac{1}{N}\sum_{i=1}^{N}\left(y^i(w^*,d_w^*) - y^i(\hat{w}^*,\hat{d}_w^*)\right)^2, \tag{36}$$

where w^* is the true optimal treatment and $\hat{w}^*$ is the optimal treatment identified by the model. The optimal treatment-dosage pair for a model is selected by first computing the optimal dosage for each treatment and then selecting the treatment with the best outcome for its optimal dosage.

Each of these metrics are computed on a held out test-set.

M Additional results

M.1 News results for source of gain

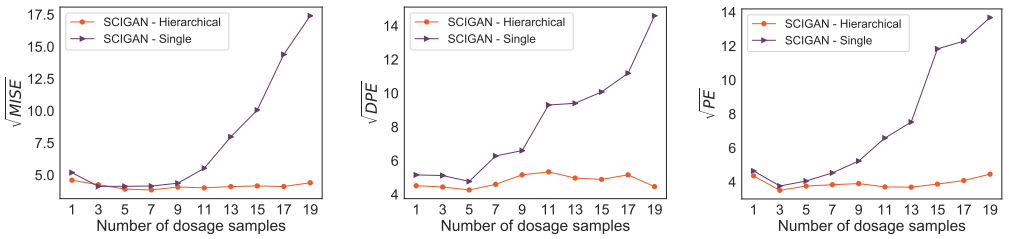
	News		
	$\sqrt{\text{MISE}}$	$\sqrt{\text{DPE}}$	$\sqrt{\text{PE}}$
Baseline	6.17 ± 0.27	6.97 ± 0.27	6.20 ± 0.21
+ $\mathcal{L}_S$	4.51 ± 0.16	4.46 ± 0.12	4.40 ± 0.11
+ Multitask	4.11 ± 0.11	4.33 ± 0.11	4.31 ± 0.11
+ Hierarchical	4.07 ± 0.05	4.24 ± 0.11	4.17 ± 0.12
+ Inv/Eqv	3.71 ± 0.05	4.14 ± 0.11	3.90 ± 0.05

Table 8: Source of gain analysis for our model on the News dataset. Metrics are reported as Mean $\pm$ Std.

M.2 Investigating hyperparameter sensitivity (n_w)

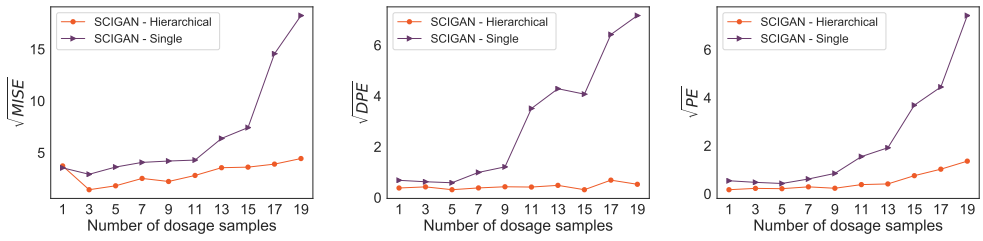
The performance of the single discriminator causes significant performance drops around $n_w = 9$ across all metrics. As previously noted, this is due to the dimension of the output space (which for $n_w = 9$ is 27) being too large. Conversely, we see that our hierarchical discriminator shows much more stable performance even when $n_w = 19$.

Here we present additional results for our investigation of the hyperparameters n_w. Fig. 8 reports each of the 3 performance metrics as we increase the number of dosage samples, n_w, used to train the discriminators on the News dataset. As with the TCGA results in the main paper we see that the single discriminator suffers a significant performance decrease when n_w is set too high.



(a) $\sqrt{\text{MISE}}$ (b) $\sqrt{\text{DPE}}$ (c) $\sqrt{\text{PE}}$

Figure 8: Performance of single vs. hierarchical discriminator when increasing the number of dosage samples (n_w) on News dataset.



(a) $\sqrt{\text{MISE}}$ (b) $\sqrt{\text{DPE}}$ (c) $\sqrt{\text{PE}}$

Figure 9: Performance of single vs. hierarchical discriminator when increasing the number of dosage samples (n_w) on TCGA dataset.

M.3 Dosage Policy Error for Benchmark Comparison

In Table 9 we report the Dosage Policy Error (DPE) corresponding to Section 6.3 in the main paper.

Methods	**TCGA** $\sqrt{\text{DPE}}$	**News** $\sqrt{\text{DPE}}$	**MIMIC** $\sqrt{\text{DPE}}$
SCIGAN	$\mathbf{0.31} \pm 0.05$	$\mathbf{4.14} \pm 0.11$	$\mathbf{0.51} \pm 0.05$
DRNet	$0.51 \pm 0.05^*$	$4.39 \pm 0.11^*$	0.52 ± 0.05
DRN-W	$0.50 \pm 0.05^*$	4.21 ± 0.11	0.53 ± 0.05
GPS	$1.38 \pm 0.01^*$	$6.40 \pm 0.01^*$	$1.41 \pm 0.12^*$
MLP-M	$0.92 \pm 0.05^*$	$4.94 \pm 0.16^*$	$0.77 \pm 0.05^*$
MLP	$1.04 \pm 0.05^*$	$5.18 \pm 0.12^*$	$0.80 \pm 0.05^*$

Table 9: Performance of individualized treatment-dose response estimation on three datasets. Bold indicates the method with the best performance for each dataset. *: performance improvement is statistically significant.

M.4 Varying the number of treatments

In this experiment, we increase the number of treatments by defining 3 or 6 additional treatments. The parameters $\mathbf{v}_1^w, \mathbf{v}_2^w, \mathbf{v}_3^w$ are defined in exactly the same way as for 3 treatments. The outcome shapes for treatments 4 and 7 are the same as for treatment 1, similarly for 5, 8 and 2 and for 6, 9 and 3. In Table 10 we report MISE, DPE and PE on the TCGA dataset with 6 treatments (TCGA-6) and with 9 treatments (TCGA-9). Note that we use 3 dosage samples for training SCIGAN in this experiment.

Method	**TCGA - 6**			**TCGA - 9**		
	$\sqrt{\text{MISE}}$	$\sqrt{\text{DPE}}$	$\sqrt{\text{PE}}$	$\sqrt{\text{MISE}}$	$\sqrt{\text{DPE}}$	$\sqrt{\text{PE}}$
SCIGAN	2.37 ± 0.12	0.43 ± 0.05	0.32 ± 0.05	2.79 ± 0.05	0.51 ± 0.05	0.54 ± 0.05
DRNET	4.09 ± 0.16	0.52 ± 0.05	0.71 ± 0.05	4.31 ± 0.12	0.59 ± 0.05	0.74 ± 0.05
GPS	6.62 ± 0.01	2.04 ± 0.01	2.61 ± 0.00	7.58 ± 0.01	3.14 ± 0.01	2.91 ± 0.01

Table 10: Performance of SCIGAN and the benchmarks when we increase the number of treatments in the dataset to 6 and 9. Bold indicates the method with the best performance for each dataset.

M.5 Sample efficiency

We have also performed a further experiment to evaluate model performance in terms of sample efficency. For the MIMIC dataset, in Table 1 we report evaluation metrics for training SCIGAN with different number of training samples N and evaluating on the same test set.

	$\sqrt{\text{MISE}}$	$\sqrt{\text{DPE}}$	$\sqrt{\text{PE}}$
$N = 100$	31.12 ± 63.39	7.72 ± 2.57	18.94 ± 29.07
$N = 500$	13.36 ± 10.46	4.07 ± 1.92	2.63 ± 0.94
$N = 1000$	3.80 ± 1.04	2.46 ± 1.75	1.03 ± 1.13
$N = 1500$	2.95 ± 0.37	0.70 ± 0.17	0.63 ± 0.12
$N = 1920$	2.09 ± 0.12	0.51 ± 0.05	0.32 ± 0.05

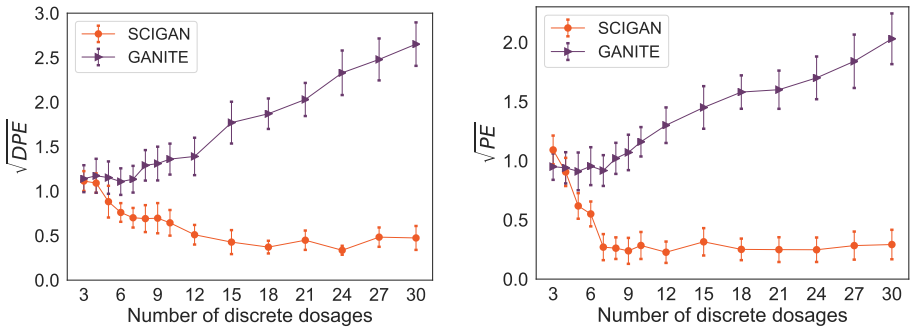
Table 11: Sample efficency analysis for MIMIC. Metrics are reported as Mean $\pm$ Std.

M.6 Discrete dosage set-up and comparison with GANITE

In this set-up, we use the TCGA dataset and treatments 2 and 3 from Table 1 with dose-response curves $f_2(\mathbf{x}, d)$ and $f_3(\mathbf{x}, d)$ respectively. Let β be the number of discrete dosages for which we want to generate data. We chose β equally spaced points in the interval $[0, 1]$ as our set of discrete dosages: $\Delta = \{\frac{k}{\beta-1}\}_{k=0}^{\beta-1}$. To create factual dosages for our dataset, we sample the dosages as before $d_w \mid x \sim \text{Beta}(\alpha, \beta_w)$ (as described in Appendix J), and choose the closest discrete dosage from the set Δ.

To evaluate SCIGAN in this setting, we maintain the same architecture for the multi-task generator and hierarchical discriminator. The only difference is that we now randomly sample dosages for the SCIGAN discriminator from Δ.

We adopt the GANITE implementation proposed by [6]. To be able to have a fair comparison with SCIGAN we also use a multi-task architecture for the GANITE generator and we give as input to each multitask head the dosage parameter. The GANITE generator will generate outcomes for all possible discrete dosages in Δ and these will be passed to the GANITE discriminator to distinguish the factual one. For the GANITE generator we use a similar architecture to the SCIGAN generator with 2 hidden layers for each multitask head and 64 neurons in each layer. The GANITE discriminator consists of 2 fully connected layers with 64 neurons in each. We also set $\lambda = 1$. In addition, to maintain a similar set-up to SCIGAN, we train an inference network to learn the counterfactual outcomes with data from the GANITE generator. The inference network has the same architecture as the GANITE generator.



(a) $\sqrt{\text{DPE}}$ (b) $\sqrt{\text{PE}}$

Figure 10: Comparison between SCIGAN and GANITE in the discrete dosage set-up.

We clearly see from Fig. 10 that SCIGAN achieves a similar performance to GANITE for a small number of dosages (< 6) but then significantly outperforms GANITE for more dosages than 6. In fact, we see that while GANITE's performance degrades with an increasing number of dosages, SCIGAN's improves and then stabilises at around 12 dosages. This is due to the fact that the single discriminator in GANITE simply cannot handle a large number of dosages. Our hierarchical model, however, can. The worse performance of SCIGAN for the lower dosages can be attributed to the fact that for such few dosages (e.g. 3 dosages corresponds to only 6 total different interventions), the SCIGAN architecture is overly complex, and the sub-sampling of dosages for the discriminator is not actually necessary.

M.7 Mixing dosage and no-dosage treatment options

We also evaluate the case when one of the treatments does not have a dosage parameter. For this experiment we generate data for the treatment that has a dosage parameter d using $f_3(\mathbf{x}, d)$ and for the treatment without an associated dosage using $2C(\mathbf{v}_0^T\mathbf{x})$, where $\mathbf{v}_0$ are parameters, $\mathbf{x}$ are patient features and $C =$ is the scaling parameter. This set-up also corresponds to the scenario where we want to compare giving a treatment with a dosage and not giving any treatment.

SCIGAN can be easily extended to incorporate an additional treatment that does not come with a dosage parameter. Such treatments will not need a dosage discriminator but will be passed to the treatment discriminator. A head can be added to the generator for each such non-dosage treatment but will not need to take dosage as an input.

As the DRNet public implementation does not allow for this set-up, we compared SCIGAN with the multilayer perceptron model with multitask heads (MLP-M). This model is trained using supervised learning to minimize error on the factual outcomes and consists of two multitask heads: one head for the treatment option which receives as input the dosage and estimates the dose-response curve and one head for the no-treatment option.

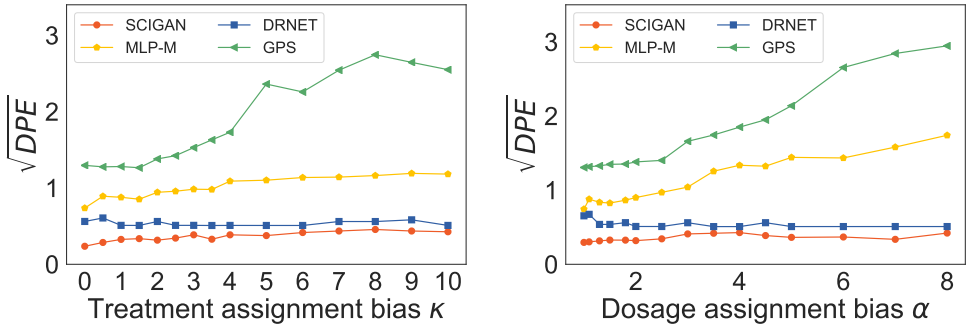
As can be seen in Table 12, SCIGAN is capable of handling this setting and lends itself naturally to potentially mixed dosage and no-dosage treatment options.

Method	TCGA			News			MIMIC		
	$\sqrt{\text{MISE}}$	$\sqrt{\text{DPE}}$	$\sqrt{\text{PE}}$	$\sqrt{\text{MISE}}$	$\sqrt{\text{DPE}}$	$\sqrt{\text{PE}}$	$\sqrt{\text{MISE}}$	$\sqrt{\text{DPE}}$	$\sqrt{\text{PE}}$
SCIGAN	**1.28** $\pm$ 0.09	**1.37** $\pm$ 0.07	**1.56** $\pm$ 0.06	**3.18** $\pm$ 0.15	**2.04** $\pm$ 0.09	**2.49** $\pm$ 0.12	**0.61** $\pm$ 0.08	**1.82** $\pm$ 0.02	**1.89** $\pm$ 0.03
MLP-M	2.08 $\pm$ 0.12	1.85 $\pm$ 0.16	2.02 $\pm$ 0.07	4.68 $\pm$ 0.11	2.45 $\pm$ 0.08	2.64 $\pm$ 0.08	1.56 $\pm$ 0.08	2.04 $\pm$ 0.03	2.14 $\pm$ 0.5

Table 12: Performance of individualized treatment-dose response when mixing treatment with no-treatment options. Bold indicates the method with the best performance for each dataset.

M.8 Additional results on selection bias

In Fig. 11 we report the DPE for our treatment and dosage bias experiment from Section 6.5 of the main paper.



(a) Treatment selection bias

(b) Dosage selection bias

Figure 11: Additional performance metrics of the 4 methods on datasets with varying bias levels on TCGA dataset

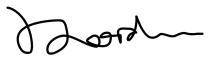
Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Estimating the Effects of Continuous-valued Interventions using Generative Adversarial Networks
Publication Status	☑ Published □ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Ioana Bica, James Jordon, and Mihaela van der Schaar. Estimating the Effects of Continuous-valued Interventions using Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, 2020.

Student Confirmation

Student Name:	James Jordon		
Contribution to the Paper	• The main idea for the paper (using hierarchical discriminators to reduce complexity of the discriminator) was the product of a discussion between Ioana and myself. • I had the idea to use Deep Sets for the hierarchical discriminators, and determined exactly how Deep Sets should be used to realise the discriminators as appropriate functions. • I conducted all of the theoretical analysis in the paper. • Ioana and I jointly wrote the entire paper through several iterations, with the theoretical section being written solely by myself and Ioana taking most of the responsibility for the experimental section. • I helped Ioana to analyse the results, and we jointly developed the semi-synthetic dataset used for the experiments. Experimental design was done jointly, with Ioana writing all of the code for the paper. • All authors contributed to the development of the paper through multiple discussions. • All authors reviewed the final draft.		
Signature 		Date	09/06/2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Professor Mihaela van der Schaar		
Supervisor comments I agree that the candidate has made a substantial contribution to the publication.		
Signature 	Date	09/06/2021

This completed form should be included in the thesis, at the end of the relevant chapter.

Chapter 3

Imputation

Publications Included

- Jinsung Yoon*, **James Jordon***, and Mihaela van der Schaar. GAIN: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning*, 2018. Weblink: `http://proceedings.mlr.press/v80/yoon18a/yoon18a.pdf`. [Accepted]

GAIN: Missing Data Imputation using Generative Adversarial Nets

Jinsung Yoon [1 *] **James Jordon** [2 *] **Mihaela van der Schaar** [1 2 3]

Abstract

We propose a novel method for imputing missing data by adapting the well-known Generative Adversarial Nets (GAN) framework. Accordingly, we call our method Generative Adversarial Imputation Nets (GAIN). The generator (G) observes some components of a real data vector, imputes the missing components conditioned on what is actually observed, and outputs a completed vector. The discriminator (D) then takes a completed vector and attempts to determine which components were actually observed and which were imputed. To ensure that D forces G to learn the desired distribution, we provide D with some additional information in the form of a *hint* vector. The hint reveals to D *partial* information about the missingness of the original sample, which is used by D to focus its attention on the imputation quality of particular components. This hint ensures that G does in fact learn to generate according to the true data distribution. We tested our method on various datasets and found that GAIN significantly outperforms state-of-the-art imputation methods.

1. Introduction

Missing data is a pervasive problem. Data may be missing because it was never collected, records were lost or for many other reasons. In the medical domain, the respiratory rate of a patient may not have been measured (perhaps because it was deemed unnecessary/unimportant) or accidentally not recorded (Yoon et al., 2017; Alaa et al., 2018). It may also be the case that certain pieces of information are difficult or even dangerous to acquire (such as information gathered from a biopsy), and so these were not gathered for those reasons (Yoon et al., 2018b). An imputation algorithm can be used to estimate missing values based on data that was observed/measured, such as the systolic blood pressure and heart rate of the patient (Yoon et al., 2018c). A substantial amount of research has been dedicated to developing imputation algorithms for medical data (Barnard & Meng, 1999; Mackinnon, 2010; Sterne et al., 2009; Purwar & Singh, 2015). Imputation algorithms are also used in many other applications such as image concealment, data compression, and counterfactual estimation (Rubin, 2004; Kreindler & Lumsden, 2012; Yoon et al., 2018a).

Missing data can be categorized into three types: (1) the data is missing completely at random (MCAR) if the missingness occurs entirely at random (there is no dependency on any of the variables), (2) the data is missing at random (MAR) if the missingness depends only on the *observed* variables[1], (3) the data is missing not at random (MNAR) if the missingness is neither MCAR nor MAR (more specifically, the data is MNAR if the missingness depends on *both observed* variables and the *unobserved* variables; thus, missingness cannot be fully accounted for by the observed variables). In this paper we provide theoretical results for our algorithm under the MCAR assumption, and compare to other state-of-the-art methods in this setting[2].

State-of-the-art imputation methods can be categorized as either discriminative or generative. Discriminative methods include MICE (Buuren & Oudshoorn, 2000; Buuren & Groothuis-Oudshoorn, 2011), MissForest (Stekhoven & Bühlmann, 2011), and matrix completion (Mazumder et al., 2010a; Yu et al., 2016; Schnabel et al., 2016; Mazumder et al., 2010b); generative methods include algorithms based on Expectation Maximization (García-Laencina et al., 2010) and algorithms based on deep learning (e.g. denoising autoencoders (DAE) and generative adversarial nets (GAN)) (Vincent et al., 2008; Gondara & Wang, 2017; Allen & Li, 2016). However, current generative methods for imputation have various drawbacks. For instance, the approach for data imputation based on (García-Laencina et al., 2010) makes assumptions about the underlying distribution and fails to generalize well when datasets contain mixed categorical and continuous variables. In contrast, the approaches based on DAE (Vincent et al., 2008) have been shown to work well in practice but require complete data during training. In

[*]Equal contribution [1]University of California, Los Angeles, CA, USA [2]University of Oxford, UK [3]Alan Turing Institute, UK. Correspondence to: Jinsung Yoon <jsyoon0823@gmail.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

[1]A formal definition of MAR can be found in the Supplementary Materials.

[2]Empirical results for the MAR and MNAR settings are shown in the Supplementary Materials.

many circumstances, missing values are part of the inherent structure of the problem so obtaining a complete dataset is impossible. Another approach with DAE (Gondara & Wang, 2017) allows for an incomplete dataset; however, it only utilizes the observed components to learn the representations of the data. (Allen & Li, 2016) uses Deep Convolutional GANs for image completion; however, it also requires complete data for training the discriminator.

In this paper, we propose a novel imputation method, which we call Generative Adversarial Imputation Nets (GAIN), that generalizes the well-known GAN (Goodfellow et al., 2014) and is able to operate successfully even when complete data is unavailable. In GAIN, the generator's goal is to accurately impute missing data, and the discriminator's goal is to distinguish between observed and imputed components. The discriminator is trained to minimize the classification loss (when classifying which components were observed and which have been imputed), and the generator is trained to maximize the discriminator's misclassification rate. Thus, these two networks are trained using an adversarial process. To achieve this goal, GAIN builds on and adapts the standard GAN architecture. To ensure that the result of this adversarial process is the desired target, the GAIN architecture provides the discriminator with additional information in the form of "hints". This hinting ensures that the generator generates samples according to the true underlying data distribution.

2. Problem Formulation

Consider a d-dimensional space $\mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_d$. Suppose that $\mathbf{X} = (X_1, ..., X_d)$ is a random variable (either continuous or binary) taking values in $\mathcal{X}$, whose distribution we will denote $P(\mathbf{X})$. Suppose that $\mathbf{M} = (M_1, ..., M_d)$ is a random variable taking values in $\{0, 1\}^d$. We will call $\mathbf{X}$ the data vector, and $\mathbf{M}$ the mask vector.

For each $i \in \{1, ..., d\}$, we define a new space $\tilde{\mathcal{X}}_i = \mathcal{X}_i \cup \{*\}$ where $*$ is simply a point not in any $\mathcal{X}_i$, representing an unobserved value. Let $\tilde{\mathcal{X}} = \tilde{\mathcal{X}}_1 \times ... \times \tilde{\mathcal{X}}_d$. We define a new random variable $\tilde{\mathbf{X}} = (\tilde{X}_1, ..., \tilde{X}_d) \in \tilde{\mathcal{X}}$ in the following way:

$$\tilde{X}_i = \begin{cases} X_i, & \text{if } M_i = 1 \\ *, & \text{otherwise} \end{cases} \quad (1)$$

so that $\mathbf{M}$ indicates which components of $\mathbf{X}$ are observed. Note that we can recover $\mathbf{M}$ from $\tilde{\mathbf{X}}$.

Throughout the remainder of the paper, we will often use lower-case letters to denote realizations of a random variable and use the notation $\mathbf{1}$ to denote a vector of 1s, whose dimension will be clear from the context (most often, d).

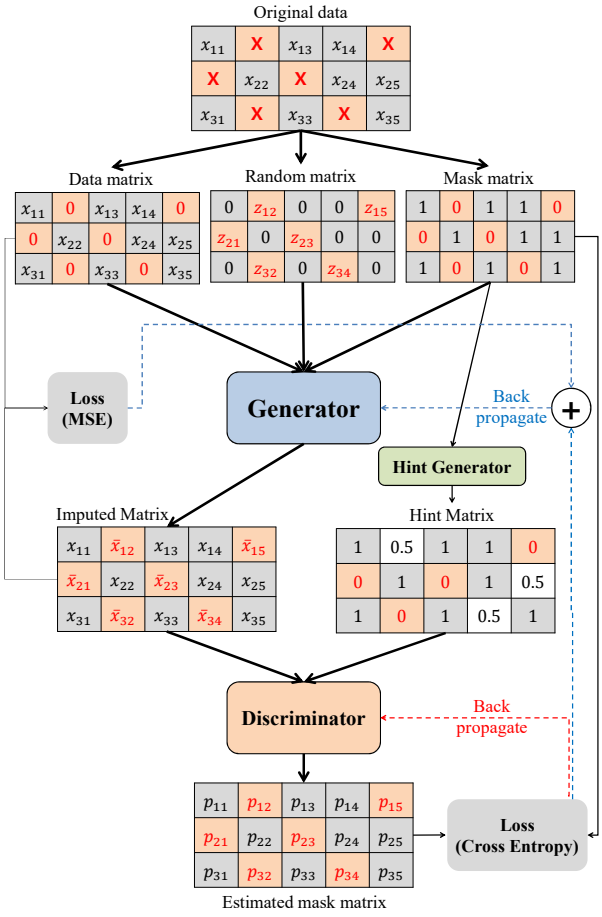


Figure 1. The architecture of GAIN

2.1. Imputation

In the imputation setting, n i.i.d. copies of $\tilde{\mathbf{X}}$ are realized, denoted $\tilde{\mathbf{x}}^1, ..., \tilde{\mathbf{x}}^n$ and we define the dataset $\mathcal{D} = \{(\tilde{\mathbf{x}}^i, \mathbf{m}^i)\}_{i=1}^n$, where $\mathbf{m}^i$ is simply the recovered realization of $\mathbf{M}$ corresponding to $\tilde{\mathbf{x}}^i$.

Our goal is to *impute* the unobserved values in each $\tilde{\mathbf{x}}_i$. Formally, we want to generate samples according to $P(\mathbf{X}|\tilde{\mathbf{X}} = \tilde{\mathbf{x}}^i)$, the conditional distribution of $\mathbf{X}$ given $\tilde{\mathbf{X}} = \tilde{\mathbf{x}}^i$, for each i, to fill in the missing data points in $\mathcal{D}$. By attempting to model the *distribution* of the data rather than just the expectation, we are able to make multiple draws and therefore make *multiple imputations* allowing us to capture the uncertainty of the imputed values (Buuren & Oudshoorn, 2000; Buuren & Groothuis-Oudshoorn, 2011; Rubin, 2004).

3. Generative Adversarial Imputation Nets

In this section we describe our approach for simulating $P(\mathbf{X}|\tilde{\mathbf{X}} = \tilde{\mathbf{x}}^i)$ which is motivated by GANs. We highlight key similarities and differences to a standard (conditional)

GAN throughout. Fig. 1 depicts the overall architecture.

3.1. Generator

The generator, G, takes (realizations of) $\tilde{\mathbf{X}}$, $\mathbf{M}$ and a noise variable, $\mathbf{Z}$, as input and outputs $\bar{\mathbf{X}}$, a vector of imputations. Let $G : \tilde{\mathcal{X}} \times \{0,1\}^d \times [0,1]^d \to \mathcal{X}$ be a function, and $\mathbf{Z} = (Z_1, ..., Z_d)$ be d-dimensional noise (independent of all other variables).

Then we define the random variables $\bar{\mathbf{X}}, \hat{\mathbf{X}} \in \mathcal{X}$ by

$$\bar{\mathbf{X}} = G(\tilde{\mathbf{X}}, \mathbf{M}, (\mathbf{1} - \mathbf{M}) \odot \mathbf{Z}) \tag{2}$$

$$\hat{\mathbf{X}} = \mathbf{M} \odot \tilde{\mathbf{X}} + (\mathbf{1} - \mathbf{M}) \odot \bar{\mathbf{X}} \tag{3}$$

where $\odot$ denotes element-wise multiplication. $\bar{\mathbf{X}}$ corresponds to the vector of *imputed* values (note that G outputs a value for every component, even if its value was observed) and $\hat{\mathbf{X}}$ corresponds to the completed data vector, that is, the vector obtained by taking the partial observation $\tilde{\mathbf{X}}$ and replacing each $*$ with the corresponding value of $\bar{\mathbf{X}}$.

This setup is very similar to a standard GAN, with $\mathbf{Z}$ being analogous to the noise variables introduced in that framework. Note, though, that in this framework, the target distribution, $P(\mathbf{X}|\tilde{\mathbf{X}})$, is essentially $||\mathbf{1} - \mathbf{M}||_1$-dimensional and so the noise we pass into the generator is $(\mathbf{1} - \mathbf{M}) \odot \mathbf{Z}$, rather than simply $\mathbf{Z}$, so that its dimension matches that of the targeted distribution.

3.2. Discriminator

As in the GAN framework, we introduce a discriminator, D, that will be used as an adversary to train G. However, unlike in a standard GAN where the output of the generator is either *completely* real or *completely* fake, in this setting the output is comprised of some components that are real and some that are fake. Rather than identifying that an entire vector is real or fake, the discriminator attempts to distinguish which *components* are real (observed) or fake (imputed) - this amounts to predicting the mask vector, $\mathbf{m}$. Note that the mask vector $\mathbf{M}$ is pre-determined by the dataset.

Formally, the discriminator is a function $D : \mathcal{X} \to [0,1]^d$ with the i-th component of $D(\hat{\mathbf{x}})$ corresponding to the probability that the i-th component of $\hat{\mathbf{x}}$ was observed.

3.3. Hint

As will be seen in the theoretical results that follow, it is necessary to introduce what we call a hint mechanism. A hint mechanism is a random variable, $\mathbf{H}$, taking values in a space $\mathcal{H}$, both of which *we define*. We allow $\mathbf{H}$ to depend on $\mathbf{M}$ and for each (imputed) sample $(\hat{\mathbf{x}}, \mathbf{m})$, we draw $\mathbf{h}$ according to the distribution $\mathbf{H}|\mathbf{M} = \mathbf{m}$. We pass $\mathbf{h}$ as an additional input to the discriminator and so it becomes a function $D : \mathcal{X} \times \mathcal{H} \to [0,1]^d$, where now the i-th component of $D(\hat{\mathbf{x}}, \mathbf{h})$ corresponds to the probability that the i-th component of $\hat{\mathbf{x}}$ was observed conditional on $\hat{\mathbf{X}} = \hat{\mathbf{x}}$ *and* $\mathbf{H} = \mathbf{h}$.

By defining $\mathbf{H}$ in different ways, we control the amount of information contained in $\mathbf{H}$ about $\mathbf{M}$ and in particular we show (in Proposition 1) that if we do not provide "enough" information about $\mathbf{M}$ to D (such as if we simply did not have a hinting mechanism), then there are several distributions that G could reproduce that would all be optimal with respect to D.

3.4. Objective

We train D to *maximize* the probability of correctly predicting $\mathbf{M}$. We train G to *minimize* the probability of D predicting $\mathbf{M}$. We define the quantity $V(D, G)$ to be

$$V(D,G) = \mathbb{E}_{\hat{\mathbf{X}},\mathbf{M},\mathbf{H}}\Big[\mathbf{M}^T \log D(\hat{\mathbf{X}}, \mathbf{H}) + (\mathbf{1} - \mathbf{M})^T \log\big(\mathbf{1} - D(\hat{\mathbf{X}}, \mathbf{H})\big)\Big], \tag{4}$$

where $\log$ is element-wise logarithm and dependence on G is through $\hat{\mathbf{X}}$.

Then, as with the standard GAN, we define the objective of GAIN to be the minimax problem given by

$$\min_G \max_D V(D,G). \tag{5}$$

We define the loss function $\mathcal{L} : \{0,1\}^d \times [0,1]^d \to \mathbb{R}$ by

$$\mathcal{L}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{d} \Big[a_i \log(b_i) + (1 - a_i) \log(1 - b_i)\Big]. \tag{6}$$

Writing $\hat{\mathbf{M}} = D(\hat{\mathbf{X}}, \mathbf{H})$, we can then rewrite (5) as

$$\min_G \max_D \mathbb{E}\big[\mathcal{L}(\mathbf{M}, \hat{\mathbf{M}})\big]. \tag{7}$$

4. Theoretical Analysis

In this section we provide a theoretical analysis of (5). Given a d-dimensional space $\mathcal{Z} = \mathcal{Z}_1 \times ... \times \mathcal{Z}_d$, a (probability) density[3] p over $\mathcal{Z}$ corresponding to a random variable Z, and a vector $\mathbf{b} \in \{0,1\}^d$ we define the set $A_\mathbf{b} = \{i : b_i = 1\}$, the projection $\phi_\mathbf{b} : \mathcal{Z} \to \Pi_{i \in A_\mathbf{b}} \mathcal{Z}_i$ by $\phi_\mathbf{b}(z) = (z_i)_{i \in A}$ and the density $p^\mathbf{b}$ to be the density of $\phi_\mathbf{b}(Z)$.

Throughout this section, we make the assumption that $\mathbf{M}$ is independent of $\mathbf{X}$, i.e. that the data is MCAR.

We will write $p(\mathbf{x}, \mathbf{m}, \mathbf{h})$ to denote the density of the random variable $(\hat{\mathbf{X}}, \mathbf{M}, \mathbf{H})$ and we will write $\hat{p}$, p_m and p_h to

[3] For ease of exposition, we use the term density even when referring to a probability mass function.

denote the marginal densities (of p) corresponding to $\hat{\mathbf{X}}$, $\mathbf{M}$ and $\mathbf{H}$, respectively. When referring to the joint density of two of the three variables (potentially conditioned on the third), we will simply use p, abusing notation slightly.

It is more intuitive to think of this density through its decomposition into densities corresponding to the true data generating process, and to the generator defined by (2),

$$\begin{aligned} p(\mathbf{x}, \mathbf{m}, \mathbf{h}) =& p_m(\mathbf{m})\hat{p}^{\mathbf{m}}(\phi_{\mathbf{m}}(\mathbf{x}|\mathbf{m})) \\ & \times \hat{p}^{\mathbf{1}-\mathbf{m}}(\phi_{\mathbf{1}-\mathbf{m}}(\mathbf{x})|\mathbf{m}, \phi_{\mathbf{m}}(\mathbf{x}))p_h(\mathbf{h}|\mathbf{m}). \end{aligned} \tag{8}$$

The first two terms in (8) are both defined by the data, where $\hat{p}^{\mathbf{m}}(\phi_{\mathbf{m}}(\mathbf{x})|\mathbf{m})$ is the density of $\phi_{\mathbf{m}}(\hat{\mathbf{X}})|\mathbf{M} = \mathbf{m}$ which corresponds to the density of $\phi_{\mathbf{m}}(\mathbf{X})$ (i.e. the true data distribution), since conditional on $\mathbf{M} = \mathbf{m}$, $\phi_{\mathbf{m}}(\hat{\mathbf{X}}) = \phi_{\mathbf{m}}(\mathbf{X})$ (see equations 1 and 3). The third term, $\hat{p}^{\mathbf{1}-\mathbf{m}}(\phi_{\mathbf{1}-\mathbf{m}}(\mathbf{x})|\mathbf{m}, \phi_{\mathbf{m}}(\mathbf{x}))$, is determined by the generator, G, and is the density of the random variable $\phi_{\mathbf{1}-\mathbf{m}}(G(\tilde{\mathbf{x}}, \mathbf{m}, \mathbf{Z})) = \phi_{\mathbf{1}-\mathbf{m}}(\bar{\mathbf{X}})|\tilde{\mathbf{X}} = \tilde{\mathbf{x}}, \mathbf{M} = \mathbf{m}$ where $\tilde{\mathbf{x}}$ is determined by $\mathbf{m}$ and $\phi_{\mathbf{m}}(\mathbf{x})$. The final term is the conditional density of the hint, which we are free to define (its selection will be motivated by the following analysis).

Using this decomposition, one can think of drawing a sample from $\hat{p}$ as first sampling $\mathbf{m}$ according to $p_m(\cdot)$, then sampling the "observed" components, $\mathbf{x}_{obs}$, according to $\hat{p}^{\mathbf{m}}(\cdot)$ (we can then construct $\tilde{\mathbf{x}}$ from $\mathbf{x}_{obs}$ and $\mathbf{m}$), then *generating* the imputed values, $\mathbf{x}_{imp}$, from the generator according to $\hat{p}^{\mathbf{1}-\mathbf{m}}(\cdot|\mathbf{m}, \mathbf{x}_{obs})$ and finally sampling the hint according to $p_h(\cdot|\mathbf{m})$.

Lemma 1. *Let $\mathbf{x} \in \mathcal{X}$. Let p_h be a fixed density over the hint space $\mathcal{H}$ and let $\mathbf{h} \in \mathcal{H}$ be such that $p(\mathbf{x}, \mathbf{h}) > 0$. Then for a fixed generator, G, the i-th component of the optimal discriminator, $D^*(\mathbf{x}, \mathbf{h})$ is given by*

$$\begin{aligned} D^*(\mathbf{x}, \mathbf{h})_i &= \frac{p(\mathbf{x}, \mathbf{h}, m_i = 1)}{p(\mathbf{x}, \mathbf{h}, m_i = 1) + p(\mathbf{x}, \mathbf{h}, m_i = 0)} && (9) \\ &= p_m(m_i = 1|\mathbf{x}, \mathbf{h}) && (10) \end{aligned}$$

for each $i \in \{1, ..., d\}$.

Proof. All proofs are provided in Supplementary Materials. □

We now rewrite (4), substituting for D^*, to obtain the following minimization criterion for G:

$$\begin{aligned} C(G) =& \mathbb{E}_{\hat{\mathbf{X}}, \mathbf{M}, \mathbf{H}}\Big(\sum_{i:M_i=1} \log p_m(m_i = 1|\hat{\mathbf{X}}, \mathbf{H}) \\ & + \sum_{i:M_i=0} \log p_m(m_i = 0|\hat{\mathbf{X}}, \mathbf{H})\Big), \end{aligned} \tag{11}$$

where dependence on G is through $p_m(\cdot|\hat{\mathbf{X}})$.

Theorem 1. *A global minimum for $C(G)$ is achieved if and only if the density $\hat{p}$ satisfies*

$$\hat{p}(\mathbf{x}|\mathbf{h}, m_i = t) = \hat{p}(\mathbf{x}|\mathbf{h}) \tag{12}$$

for each $i \in \{1, ..., d\}$, $\mathbf{x} \in \mathcal{X}$ and $\mathbf{h} \in \mathcal{H}$ such that $p_h(\mathbf{h}|m_i = t) > 0$.

The following proposition asserts that if $\mathbf{H}$ does not contain "enough" information about $\mathbf{M}$, we cannot guarantee that G learns the desired distribution (the one uniquely defined by the (underlying) data).

Proposition 1. *There exist distributions of $\mathbf{X}$, $\mathbf{M}$ and $\mathbf{H}$ for which solutions to (12) are not unique. In fact, if $\mathbf{H}$ is independent of $\mathbf{M}$, then (12) does not define a unique density, in general.*

Let the random variable $\mathbf{B} = (B_1, ..., B_d) \in \{0, 1\}^d$ be defined by first sampling k from $\{1, ..., d\}$ uniformly at random and then setting

$$B_j = \begin{cases} 1 \text{ if } j \neq k \\ 0 \text{ if } j = k. \end{cases} \tag{13}$$

Let $\mathcal{H} = \{0, 0.5, 1\}^d$ and, given $\mathbf{M}$, define

$$\mathbf{H} = \mathbf{B} \odot \mathbf{M} + 0.5(\mathbf{1} - \mathbf{B}). \tag{14}$$

Observe first that $\mathbf{H}$ is such that $H_i = t \implies M_i = t$ for $t \in \{0, 1\}$ but that $H_i = 0.5$ implies nothing about M_i. In other words, $\mathbf{H}$ reveals all but one of the components of $\mathbf{M}$ to D. Note, however, that $\mathbf{H}$ does contain some information about M_i since M_i is not assumed to be independent of the other components of $\mathbf{M}$.

The following lemma confirms that the discriminator behaves as we expect with respect to this hint mechanism.

Lemma 2. *Suppose $\mathbf{H}$ is defined as above. Then for $\mathbf{h}$ such that $h_i = 0$ we have $D^*(\mathbf{x}, \mathbf{h})_i = 0$ and for $\mathbf{h}$ such that $h_i = 1$ we have $D^*(\mathbf{x}, \mathbf{h})_i = 1$, for all $\mathbf{x} \in \mathcal{X}$, $i \in \{1, ..., d\}$.*

The final proposition we state tells us that $\mathbf{H}$ as specified above ensures the generator learns to replicate the desired distribution.

Proposition 2. *Suppose $\mathbf{H}$ is defined as above. Then the solution to (12) is unique and satisfies*

$$\hat{p}(\mathbf{x}|\mathbf{m}_1) = \hat{p}(\mathbf{x}|\mathbf{m}_2) \tag{15}$$

for all $\mathbf{m}_1, \mathbf{m}_2 \in \{0, 1\}^d$. In particular, $\hat{p}(\mathbf{x}|\mathbf{m}) = \hat{p}(\mathbf{x}|\mathbf{1})$ and since $\mathbf{M}$ is independent of $\mathbf{X}$, $\hat{p}(\mathbf{x}|\mathbf{1})$ is the density of $\mathbf{X}$. The distribution of $\hat{\mathbf{X}}$ is therefore the same as the distribution of $\mathbf{X}$.

For the remainder of the paper, $\mathbf{B}$ and $\mathbf{H}$ will be defined as in equations (13) and (14).

5. GAIN Algorithm

Using an approach similar to that in (Goodfellow et al., 2014), we solve the minimax optimization problem (5) in an iterative manner. Both G and D are modeled as fully connected neural nets.

We first optimize the discriminator D with a fixed generator G using mini-batches of size k_D[4]. For each sample in the mini-batch, $(\tilde{\mathbf{x}}(j), \mathbf{m}(j))$[5], we draw k_D independent samples, $\mathbf{z}(j)$ and $\mathbf{b}(j)$, of $\mathbf{Z}$ and $\mathbf{B}$ and compute $\hat{\mathbf{x}}(j)$ and $\mathbf{h}(j)$ accordingly. Lemma 2 then tells us that the only outputs of D that depend on G are the ones corresponding to $b_i = 0$ for each sample. We therefore only train D to give us these outputs (if we also trained D to match the outputs specified in Lemma 2 we would gain no information about G, but D would overfit to the hint vector). We define $\mathcal{L}_D : \{0,1\}^d \times [0,1]^d \times \{0,1\}^d \to \mathbb{R}$ by

$$\mathcal{L}_D(\mathbf{m}, \hat{\mathbf{m}}, \mathbf{b}) = \sum_{i:b_i=0} \Big[m_i \log(\hat{m}_i) + (1-m_i)\log(1-\hat{m}_i) \Big]. \tag{16}$$

D is then trained according to

$$\min_D - \sum_{j=1}^{k_D} \mathcal{L}_D(\mathbf{m}(j), \hat{\mathbf{m}}(j), \mathbf{b}(j)) \tag{17}$$

recalling that $\hat{\mathbf{m}}(j) = D(\hat{\mathbf{x}}(j), \mathbf{m}(j))$.

Second, we optimize the generator G using the newly updated discriminator D with mini-batches of size k_G. We first note that G in fact outputs a value for the *entire* data vector (including values for the components we observed). Therefore, in training G, we not only ensure that the imputed values for missing components ($m_j = 0$) successfully fool the discriminator (as defined by the minimax game), we also ensure that the values outputted by G for observed components ($m_j = 1$) are close to those actually observed. This is justified by noting that the conditional distribution of $\mathbf{X}$ given $\tilde{\mathbf{X}} = \tilde{\mathbf{x}}$ obviously fixes the components of $\mathbf{X}$ corresponding to $M_i = 1$ to be $\tilde{X}_i$. This also ensures that the representations learned in the hidden layers of $\tilde{\mathbf{X}}$ suitably capture the information contained in $\tilde{\mathbf{X}}$ (as in an auto-encoder).

To achieve this, we define two different loss functions. The first, $\mathcal{L}_G : \{0,1\}^d \times [0,1]^d \times \{0,1\}^d \to \mathbb{R}$, is given by

$$\mathcal{L}_G(\mathbf{m}, \hat{\mathbf{m}}, \mathbf{b}) = - \sum_{i:b_i=0} (1-m_i)\log(\hat{m}_i), \tag{18}$$

[4]Details of hyper-parameter selection can be found in the Supplementary Materials.

[5]The index j now corresponds to the j-th sample of the mini-batch, rather than the j-th sample of the entire dataset.

Algorithm 1 Pseudo-code of GAIN

while training loss has not converged **do**

(1) Discriminator optimization

Draw k_D samples from the dataset $\{(\tilde{\mathbf{x}}(j), \mathbf{m}(j))\}_{j=1}^{k_D}$

Draw k_D i.i.d. samples, $\{\mathbf{z}(j)\}_{j=1}^{k_D}$, of $\mathbf{Z}$

Draw k_D i.i.d. samples, $\{\mathbf{b}(j)\}_{j=1}^{k_D}$, of $\mathbf{B}$

for $j = 1, ..., k_D$ **do**

$\bar{\mathbf{x}}(j) \leftarrow G(\tilde{\mathbf{x}}(j), \mathbf{m}(j), \mathbf{z}(j))$

$\hat{\mathbf{x}}(j) \leftarrow \mathbf{m}(j) \odot \tilde{\mathbf{x}}(j) + (\mathbf{1} - \mathbf{m}(j)) \odot \bar{\mathbf{x}}(j)$

$\mathbf{h}(j) = \mathbf{b}(j) \odot \mathbf{m}(j) + 0.5(\mathbf{1} - \mathbf{b}(j))$

end for

Update D using stochastic gradient descent (SGD)

$$\nabla_D - \sum_{j=1}^{k_D} \mathcal{L}_D(\mathbf{m}(j), D(\hat{\mathbf{x}}(j), \mathbf{h}(j)), \mathbf{b}(j))$$

(2) Generator optimization

Draw k_G samples from the dataset $\{(\tilde{\mathbf{x}}(j), \mathbf{m}(j))\}_{j=1}^{k_G}$

Draw k_G i.i.d. samples, $\{\mathbf{z}(j)\}_{j=1}^{k_G}$ of $\mathbf{Z}$

Draw k_G i.i.d. samples, $\{\mathbf{b}(j)\}_{j=1}$ of $\mathbf{B}$

for $j = 1, ..., k_G$ **do**

$\mathbf{h}(j) = \mathbf{b}(j) \odot \mathbf{m}(j) + 0.5(\mathbf{1} - \mathbf{b}(j))$

end for

Update G using SGD (for fixed D)

$$\nabla_G \sum_{j=1}^{k_G} \mathcal{L}_G(\mathbf{m}(j), \hat{\mathbf{m}}(j), \mathbf{b}(j)) + \alpha \mathcal{L}_M(\mathbf{x}(j), \tilde{\mathbf{x}}(j))$$

end while

and the second, $\mathcal{L}_M : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, by

$$\mathcal{L}_M(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} m_i L_M(x_i, x_i'), \tag{19}$$

where

$$L_M(x_i, x_i') = \begin{cases} (x_i' - x_i)^2, & \text{if } x_i \text{ is continuous,} \\ -x_i \log(x_i'), & \text{if } x_i \text{ is binary.} \end{cases}$$

As can be seen from their definitions, $\mathcal{L}_G$ will apply to the missing components ($m_i = 0$) and $\mathcal{L}_M$ will apply to the observed components ($m_i = 1$).

$\mathcal{L}_G(\mathbf{m}, \hat{\mathbf{m}})$ is smaller when $\hat{m}_i$ is closer to 1 for i such that $m_i = 0$. That is, $\mathcal{L}_G(\mathbf{m}, \hat{\mathbf{m}})$ is smaller when D is less able to identify the imputed values as being imputed (it falsely categorizes them as observed). $\mathcal{L}_M(\mathbf{x}, \tilde{\mathbf{x}})$ is minimized when the reconstructed features (i.e. the values G outputs for features that were observed) are close to the actually observed features.

G is then trained to minimize the weighted sum of the two

Table 1. Source of gains in GAIN algorithm (Mean ± Std of RMSE (Gain (%)))

Algorithm	Breast	Spam	Letter	Credit	News
GAIN	**.0546 ± .0006**	**.0513± .0016**	**.1198± .0005**	**.1858 ± .0010**	**.1441 ± .0007**
GAIN w/o $\mathcal{L}_G$	.0701 ± .0021 **(22.1%)**	.0676 ± .0029 **(24.1%)**	.1344 ± .0012 **(10.9%)**	.2436 ± .0012 **(23.7%)**	.1612 ± .0024 **(10.6%)**
GAIN w/o $\mathcal{L}_M$	.0767 ± .0015 **(28.9%)**	.0672 ± .0036 **(23.7%)**	.1586 ± .0024 **(24.4%)**	.2533 ± .0048 **(26.7%)**	.2522 ± .0042 **(42.9%)**
GAIN w/o Hint	.0639 ± .0018 **(14.6%)**	.0582 ± .0008 **(11.9%)**	.1249 ± .0011 **(4.1%)**	.2173 ± .0052 **(14.5%)**	.1521 ± .0008 **(5.3%)**
GAIN w/o Hint & $\mathcal{L}_M$	.0782 ± .0016 **(30.1%)**	.0700 ± .0064 **(26.7%)**	.1671 ± .0052 **(28.3%)**	.2789 ± .0071 **(33.4%)**	.2527 ± .0052 **(43.0%)**

losses as follows:

$$\min_G \sum_{j=1}^{k_G} \mathcal{L}_G(\mathbf{m}(j), \hat{\mathbf{m}}(j), \mathbf{b}(j)) + \alpha \mathcal{L}_M(\tilde{\mathbf{x}}(j), \hat{\mathbf{x}}(j)),$$

where α is a hyper-parameter.

The pseudo-code is presented in Algorithm 1.

6. Experiments

In this section, we validate the performance of GAIN using multiple real-world datasets. In the first set of experiments we qualitatively analyze the properties of GAIN. In the second we quantitatively evaluate the imputation performance of GAIN using various UCI datasets (Lichman, 2013), giving comparisons with state-of-the-art imputation methods. In the third we evaluate the performance of GAIN in various settings (such as on datasets with different missing rates). In the final set of experiments we evaluate GAIN against other imputation algorithms when the goal is to perform prediction on the imputed dataset.

We conduct each experiment 10 times and within each experiment we use 5-cross validations. We report either RMSE or AUROC as the performance metric along with their standard deviations across the 10 experiments. Unless otherwise stated, missingness is applied to the datasets by randomly removing 20% of all data points (MCAR).

6.1. Source of gain

The potential sources of gain for the GAIN framework are: the use of a GAN-like architecture (through $\mathcal{L}_G$), the use of reconstruction error in the loss ($\mathcal{L}_M$), and the use of the hint ($\mathbf{H}$). In order to understand how each of these affects the performance of GAIN, we exclude one or two of them and compare the performances of the resulting architectures against the full GAIN architecture.

Table 1 shows that the performance of GAIN is improved when all three components are included. More specifically, the full GAIN framework has a 15% improvement over the simple auto-encoder model (i.e. GAIN w/o $\mathcal{L}_G$). Furthermore, utilizing the hint vector additionally gives improvements of 10%.

6.2. Quantitative analysis of GAIN

We use five real-world datasets from UCI Machine Learning Repository (Lichman, 2013) (Breast, Spam, Letter, Credit, and News) to quantitatively evaluate the imputation performance of GAIN. Details of each dataset can be found in the Supplementary Materials.

In table 2 we report the RMSE (and its standard deviation) for GAIN and 5 other state-of-the-art imputation methods: MICE (Buuren & Oudshoorn, 2000; Buuren & Groothuis-Oudshoorn, 2011), MissForest (Stekhoven & Bühlmann, 2011), Matrix completion (Matrix) (Mazumder et al., 2010a), Auto-encoder (Gondara & Wang, 2017) and Expectation-maximization (EM) (García-Laencina et al., 2010). As can be seen from the table, GAIN significantly outperforms each benchmark. Results for the imputation quality of categorical variables in this experiment are given in the Supplementary Materials.

6.3. GAIN in different settings

To better understand GAIN, we conduct several experiments in which we vary the missing rate, the number of samples, and the number of dimensions using Credit dataset. Fig. 2 shows the performance (RMSE) of GAIN within these different settings in comparison to the two most competi-

Table 2. Imputation performance in terms of RMSE (Average ± Std of RMSE)

Algorithm	Breast	Spam	Letter	Credit	News
GAIN	**.0546 ± .0006**	**.0513± .0016**	**.1198± .0005**	**.1858 ± .0010**	**.1441 ± .0007**
MICE	.0646 ± .0028	.0699 ± .0010	.1537 ± .0006	.2585 ± .0011	.1763 ± .0007
MissForest	.0608 ± .0013	.0553 ± .0013	.1605 ± .0004	.1976 ± .0015	.1623 ± 0.012
Matrix	.0946 ± .0020	.0542 ± .0006	.1442 ± .0006	.2602 ± .0073	.2282 ± .0005
Auto-encoder	.0697 ± .0018	.0670 ± .0030	.1351 ± .0009	.2388 ± .0005	.1667 ± .0014
EM	.0634 ± .0021	.0712 ± .0012	.1563 ± .0012	.2604 ± .0015	.1912 ± .0011

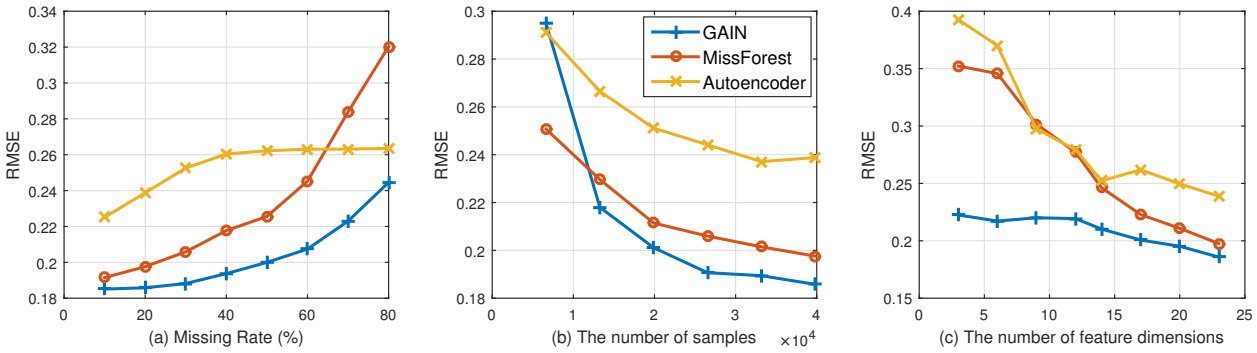


Figure 2. RMSE performance in different settings: (a) Various missing rates, (b) Various number of samples, (c) Various feature dimensions

tive benchmarks (MissForest and Auto-encoder). Fig. 2 (a) shows that, even though the performance of each algorithm decreases as missing rates increase, GAIN consistently outperforms the benchmarks across the entire range of missing rates.

Fig. 2 (b) shows that as the number of samples increases, the performance improvements of GAIN over the benchmarks also increases. This is due to the large number of parameters in GAIN that need to be optimized, however, as demonstrated on the Breast dataset (in Table 2), GAIN is still able to outperform the benchmarks even when the number of samples is relatively small.

Fig. 2 (c) shows that GAIN is also robust to the number of feature dimensions. On the other hand, the discriminative model (MissForest) cannot as easily cope when the number of feature dimensions is small.

6.4. Prediction Performance

We now compare GAIN against the same benchmarks with respect to the accuracy of post-imputation prediction. For this purpose, we use Area Under the Receiver Operating Characteristic Curve (AUROC) as the measure of performance. To be fair to all methods, we use the same predictive model (logistic regression) in all cases.

Comparisons are made on all datasets except Letter (as it has multi-class labels) and the results are reported in Table 3.

As Table 3 shows, GAIN, which we have already shown to achieve the best imputation accuracy (in Table 2), yields the best post-imputation prediction accuracy. However, even in cases where the improvement in imputation accuracy is large, the improvements in prediction accuracy are not always significant. This is probably due to the fact that there is sufficient information in the (80%) observed data to predict the label.

Prediction accuracy with various missing rates: In this experiment, we evaluate the post-imputation prediction performance when the missing rate of the dataset is varied. Note that every dataset (except Letter) has their own binary label.

The results of this experiment (for GAIN and the two most competitive benchmarks) are shown in Fig. 3. In particular, the performance of GAIN is significantly better than the other two for higher missing rates, this is due to the fact that as the information contained in the observed data decreases (due to more values being missing), the imputation quality becomes more important, and GAIN has already been shown to provide (significantly) better quality imputations.

Table 3. Prediction performance comparison

Algorithm	AUROC (Average ± Std)			
	Breast	Spam	Credit	News
GAIN	**.9930 ± .0073**	**.9529 ± .0023**	**.7527 ± .0031**	**.9711 ± .0027**
MICE	.9914 ± .0034	.9495 ± .0031	.7427 ± .0026	.9451 ± .0037
MissForest	.9860 ± .0112	.9520 ± .0061	.7498 ± .0047	.9597 ± .0043
Matrix	.9897 ± .0042	.8639 ± .0055	.7059 ± .0150	.8578 ± .0125
Auto-encoder	.9916 ± .0059	.9403 ± .0051	.7485 ± .0031	.9321 ± .0058
EM	.9899 ± .0147	.9217 ± .0093	.7390 ± .0079	.8987 ± .0157

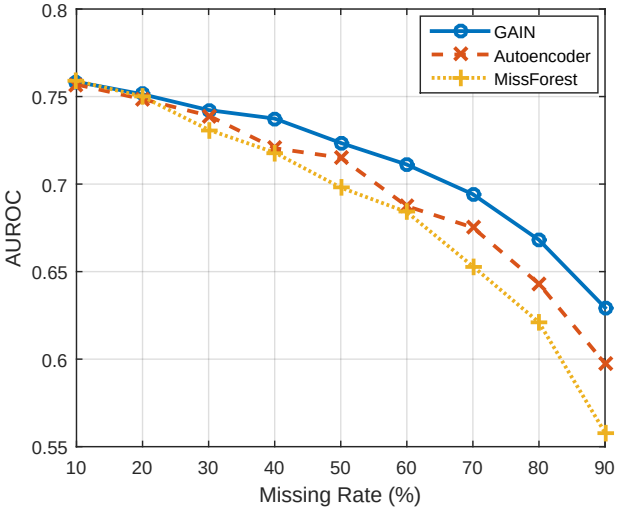


Figure 3. The AUROC performance with various missing rates with Credit dataset

6.5. Congeniality of GAIN

The congeniality of an imputation model is its ability to impute values that respect the feature-label relationship (Meng, 1994; Burgess et al., 2013; Deng et al., 2016). The congeniality of an imputation model can be evaluated by measuring the effects on the feature-label relationships after the imputation. We compare the logistic regression parameters, $\mathbf{w}$, learned from the complete Credit dataset with the parameters, $\hat{\mathbf{w}}$, learned from an incomplete Credit dataset by first imputing and then performing logistic regression.

We report the mean and standard deviation of both the mean bias ($||\mathbf{w} - \hat{\mathbf{w}}||_1$) and the mean square error ($||\mathbf{w} - \hat{\mathbf{w}}||_2$) for each method in Table 4. These quantities being lower indicates that the imputation algorithm better respects the relationship between feature and label. As can be seen in the table, GAIN achieves significantly lower mean bias and mean square error than other state-of-the-art imputation algorithms (from 8.9% to 79.2% performance improvements).

Table 4. Congeniality of imputation models

Algorithm	Mean Bias ($\|\|\mathbf{w} - \hat{\mathbf{w}}\|\|_1$)	MSE ($\|\|\mathbf{w} - \hat{\mathbf{w}}\|\|_2$)
GAIN	**0.3163± 0.0887**	**0.5078± 0.1137**
MICE	0.8315 ± 0.2293	0.9467 ± 0.2083
MissForest	0.6730 ± 0.1937	0.7081 ± 0.1625
Matrix	1.5321 ± 0.0017	1.6660 ± 0.0015
Auto-encoder	0.3500 ± 0.1503	0.5608 ±0.1697
EM	0.8418 ± 0.2675	0.9369 ± 0.2296

7. Conclusion

We propose a generative model for missing data imputation, GAIN. This novel architecture generalizes the well-known GAN such that it can deal with the unique characteristics of the imputation problem. Various experiments with real-world datasets show that GAIN significantly outperforms state-of-the-art imputation techniques. The development of a new, state-of-the-art technique for imputation can have transformative impacts; most datasets in medicine as well as in other domains have missing data. Future work will investigate the performance of GAIN in recommender systems, error concealment as well as in active sensing (Yu et al., 2009). Preliminary results in error concealment using the MNIST dataset (LeCun & Cortes, 2010) can be found in the Supplementary Materials - see Fig. 4 and 5.

Acknowledgement

The authors would like to thank the reviewers for their helpful comments. The research presented in this paper was supported by the Office of Naval Research (ONR) and the NSF (Grant number: ECCS1462245, ECCS1533983, and ECCS1407712).

References

Alaa, A. M., Yoon, J., Hu, S., and van der Schaar, M. Personalized risk scoring for critical care prognosis using mixtures of gaussian processes. *IEEE Transactions on Biomedical Engineering*, 65(1):207–218, 2018.

Allen, A. and Li, W. *Generative Adversarial Denoising Autoencoder for Face Completion*, 2016. URL https://www.cc.gatech.edu/~hays/7476/projects/Avery_Wenchen/.

Barnard, J. and Meng, X.-L. Applications of multiple imputation in medical studies: from aids to nhanes. *Statistical methods in medical research*, 8(1):17–36, 1999.

Burgess, S., White, I. R., Resche-Rigon, M., and Wood, A. M. Combining multiple imputation and meta-analysis with individual participant data. *Statistics in medicine*, 32(26):4499–4514, 2013.

Buuren, S. and Groothuis-Oudshoorn, K. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45(3), 2011.

Buuren, S. v. and Oudshoorn, C. Multivariate imputation by chained equations: Mice v1. 0 user's manual. Technical report, TNO, 2000.

Deng, Y., Chang, C., Ido, M. S., and Long, Q. Multiple imputation for general missing data patterns in the presence of high-dimensional data. *Scientific reports*, 6:21689, 2016.

García-Laencina, P. J., Sancho-Gómez, J.-L., and Figueiras-Vidal, A. R. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19(2): 263–282, 2010.

Gondara, L. and Wang, K. Multiple imputation using deep denoising autoencoders. *arXiv preprint arXiv:1705.02737*, 2017.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural information processing systems*, pp. 2672–2680, 2014.

Kreindler, D. M. and Lumsden, C. J. The effects of the irregular sample and missing data in time series analysis. *Nonlinear Dynamical Systems Analysis for the Behavioral Sciences Using Real Data*, pp. 135, 2012.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Mackinnon, A. The use and reporting of multiple imputation in medical research–a review. *Journal of internal medicine*, 268(6):586–593, 2010.

Mazumder, R., Hastie, T., and Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug): 2287–2322, 2010a.

Mazumder, R., Hastie, T., and Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug): 2287–2322, 2010b.

Meng, X.-L. Multiple-imputation inferences with uncongenial sources of input. *Statistical Science*, pp. 538–558, 1994.

Purwar, A. and Singh, S. K. Hybrid prediction model with missing value imputation for medical data. *Expert Systems with Applications*, 42(13):5621–5631, 2015.

Rubin, D. B. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

Schnabel, T., Swaminatan, A., Singh, A., Chandak, N., and Joachims, T. Recommendations as treatments: debiasing learning and evolution. *ICML*, 2016.

Stekhoven, D. J. and Bühlmann, P. Missforestnonparametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2011.

Sterne, J. A., White, I. R., Carlin, J. B., Spratt, M., Royston, P., Kenward, M. G., Wood, A. M., and Carpenter, J. R. Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *BMJ*, 338: b2393, 2009.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International conference on Machine learning*, pp. 1096–1103. ACM, 2008.

Yoon, J., Davtyan, C., and van der Schaar, M. Discovery and clinical decision support for personalized healthcare. *IEEE journal of biomedical and health informatics*, 21 (4):1133–1145, 2017.

Yoon, J., Jordon, J., and van der Schaar, M. GANITE: Estimation of individualized treatment effects using generative adversarial nets. In *International Conference on Learning Representations*, 2018a. URL https://openreview.net/forum?id=ByKWUeWA-.

Yoon, J., Zame, W. R., Banerjee, A., Cadeiras, M., Alaa, A. M., and van der Schaar, M. Personalized survival predictions via trees of predictors: An application to cardiac transplantation. *PloS one*, 13(3):e0194985, 2018b.

Yoon, J., Zame, W. R., and van der Schaar, M. Deep sensing: Active sensing using multi-directional recurrent neural networks. In *International Conference on Learning Representations*, 2018c. URL https://openreview.net/forum?id=r1SnX5xCb.

Yu, H.-F., Rao, H., and Dhillon, I. S. Temporal regularized matrix factorization for high-dimensional time series prediction. *NIPS*, 2016.

Yu, S., Krishnapuram, B., Rosales, R., and Rao, R. B. Active sensing. In *Artificial Intelligence and Statistics*, pp. 639–646, 2009.

Supplementary Materials
GAIN: Missing Data Imputation using Generative Adversarial Nets

Jinsung Yoon [1 *] **James Jordon** [2 *] **Mihaela van der Schaar** [1 2 3]

1. Proofs

1.1. Proof of Lemma 1

For $t \in \{0,1\}$ define the set $M_t^i = \{\mathbf{m} \in \{0,1\}^d : m_i = t\}$.

$$
\begin{aligned}
V(D,G) &= \mathbb{E}_{\tilde{\mathbf{X}},\mathbf{z},\mathbf{M},\mathbf{H}}\Big[\mathbf{M}^T \log D\big(G(\tilde{\mathbf{X}},\mathbf{M}),\mathbf{H}\big) + (\mathbf{1}-\mathbf{M})^T \log\Big(\mathbf{1} - D\big(G(\tilde{\mathbf{X}},\mathbf{M}),\mathbf{H}\big)\Big)\Big] \\
&= \mathbb{E}_{\hat{\mathbf{X}},\mathbf{M},\mathbf{H}}\Big[\mathbf{M}^T \log D(\hat{\mathbf{X}},\mathbf{H}) + (\mathbf{1}-\mathbf{M})^T \log\big(\mathbf{1} - D(\hat{\mathbf{X}},\mathbf{H})\big)\Big] \\
&= \int_{\mathcal{X}} \sum_{\mathbf{m}\in\{0,1\}^d} \int_{\mathcal{H}} \left(\mathbf{m}^T \log D(\mathbf{x},\mathbf{h}) + (\mathbf{1}-\mathbf{m})^T \log(\mathbf{1} - D(\mathbf{x},\mathbf{h}))\right) p(\mathbf{x},\mathbf{m},\mathbf{h}) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \int_{\mathcal{X}} \int_{\mathcal{H}} \sum_{\mathbf{m}\in\{0,1\}^d} \left(\sum_{i:m_i=1} \log D(\mathbf{x},\mathbf{h})_i + \sum_{i:m_i=0} \log(\mathbf{1} - D(\mathbf{x},\mathbf{h})_i)\right) p(\mathbf{x},\mathbf{m},\mathbf{h}) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&\overset{(1)}{=} \int_{\mathcal{X}} \int_{\mathcal{H}} \sum_{i=1}^{d} \left(\sum_{\mathbf{m}\in M_1^i} \log D(\mathbf{x},\mathbf{h})_i + \sum_{\mathbf{m}\in M_0^i} \log(\mathbf{1} - D(\mathbf{x},\mathbf{h})_i)\right) p(\mathbf{x},\mathbf{m},\mathbf{h}) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \int_{\mathcal{X}} \int_{\mathcal{H}} \sum_{i=1}^{d} \left(\log D(\mathbf{x},\mathbf{h})_i \sum_{\mathbf{m}\in M_1^i} p(\mathbf{x},\mathbf{m},\mathbf{h})\right) + \left(\log(\mathbf{1} - D(\mathbf{x},\mathbf{h})_i) \sum_{\mathbf{m}\in M_0^i} p(\mathbf{x},\mathbf{m},\mathbf{h})\right) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \int_{\mathcal{X}} \int_{\mathcal{H}} \sum_{i=1}^{d} \log D(\mathbf{x},\mathbf{h})_i p(\mathbf{x},\mathbf{h},m_i=1) + \log(\mathbf{1} - D(\mathbf{x},\mathbf{h})_i) p(\mathbf{x},\mathbf{h},m_i=0) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x}
\end{aligned}
$$

where (1) follows from switching the order of summation. We then note that $y \mapsto a\log y + b\log(1-y)$ achieves its maximum in $[0,1]$ at $\frac{a}{a+b}$ and so $V(D,G)$ is maximized (for fixed G) when

$$
D(\mathbf{x},\mathbf{h})_i = \frac{p(\mathbf{x},\mathbf{h},m_i=1)}{p(\mathbf{x},\mathbf{h},m_i=0) + p(\mathbf{x},\mathbf{h},m_i=1)} \tag{1}
$$

for each $i \in \{1,...,d\}$. □.

[*]Equal contribution [1]University of California, Los Angeles, CA, USA [2]University of Oxford, UK [3]Alan Turing Institute, UK. Correspondence to: Jinsung Yoon <jsyoon0823@gmail.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

1.2. Proof of Theorem 1

Denote by $\mathcal{H}_t^i$ the space $\{\mathbf{h} \in \mathcal{H} : p_h(\mathbf{h}|m_i = t) > 0\}$.

$$
\begin{aligned}
C(G) &= \mathbb{E}_{\hat{\mathbf{X}},\mathbf{M},\mathbf{H}}\Big(\sum_{i:M_i=1} \log p_m(m_i = 1|\hat{\mathbf{X}}, \mathbf{H}) + \sum_{i:M_i=0} \log p_m(m_i = 0|\hat{\mathbf{X}}, \mathbf{H})\Big) \\
&= \int_{\mathcal{X}} \int_{\mathcal{H}} \sum_{i=1}^{d} \big[p(\mathbf{x}, \mathbf{h}, m_i = 1) \log p_m(m_i = 1|\mathbf{x}, \mathbf{h}) + p(\mathbf{x}, \mathbf{h}, m_i = 0) \log p_m(m_i = 0|\mathbf{x}, \mathbf{h})\big] \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_i^i} \int_{\mathcal{X}} p(\mathbf{x}, \mathbf{h}, m_i = t) \log p_m(m_i = t|\mathbf{x}, \mathbf{h}) \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_i^i} \int_{\mathcal{X}} p(\mathbf{x}, \mathbf{h}, m_i = t) \log \frac{p(\mathbf{x}, m_i = t|\mathbf{h})}{\hat{p}(\mathbf{x}|\mathbf{h})} \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&= \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_i^i} \int_{\mathcal{X}} p(\mathbf{x}, \mathbf{h}, m_i = t) \log \frac{p(\mathbf{x}, m_i = t|\mathbf{h}) p_m(m_i = t|\mathbf{h})}{\hat{p}(\mathbf{x}|\mathbf{h}) p_m(m_i = t|\mathbf{h})} \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&\overset{(2)}{=} \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} \int_{\mathcal{X}} p(\mathbf{x}, \mathbf{h}, m_i = t) \log \frac{\hat{p}(\mathbf{x}|\mathbf{h}, m_i = t)}{\hat{p}(\mathbf{x}|\mathbf{h})} \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} + \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} p(m_i = t, \mathbf{h}) \log p_m(m_i = t|\mathbf{h}) \mathrm{d}\mathbf{h} \\
&= \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} \int_{\mathcal{X}} p(m_i = t, \mathbf{h}) \hat{p}(\mathbf{x}|\mathbf{h}, m_i = t) \log \frac{\hat{p}(\mathbf{x}|\mathbf{h}, m_i = t)}{\hat{p}(\mathbf{x}|\mathbf{h})} \mathrm{d}\mathbf{h}\mathrm{d}\mathbf{x} \\
&\qquad + \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} p(m_i = t, \mathbf{h}) \log p_m(m_i = t|\mathbf{h}) \mathrm{d}\mathbf{h} \\
&= \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} p_m(m_i = t, \mathbf{h}) D_{\mathrm{KL}}(\hat{p}(\cdot|\mathbf{h}, m_i = t)||\hat{p}(\cdot|\mathbf{h})) + \sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} p(m_i = t, \mathbf{h}) \log p_m(m_i = t|\mathbf{h}) \mathrm{d}\mathbf{h}
\end{aligned}
$$

where (2) follows by using the identity $\log(ab) = \log(a) + \log(b)$ and integrating out $\mathbf{x}$ from the resulting second term.

We then note that for any two densities, p and q, the Kullback-Leibler divergence $D_{\mathrm{KL}}(p||q)$ of p with respect to q is 0 if and only if $p = q$ (almost everywhere).

It follows that C(G) is minimized if only if for every $i \in \{1, ..., d\}$, $t \in \{0, 1\}$, $\mathbf{h} \in \mathcal{H}_t^i$ we have that for (almost) every $\mathbf{x}$, $\hat{p}(\mathbf{x}|\mathbf{h}, m_i = t) = \hat{p}(\mathbf{x}|\mathbf{h})$.

The minimum value is then given by $\sum_{i=1}^{d} \sum_{t\in\{0,1\}} \int_{\mathcal{H}_t^i} p(m_i = t, \mathbf{h}) \log p_m(m_i = t|\mathbf{h}) \mathrm{d}\mathbf{h}$, which lines up with the intuition that the worst a discriminator can (always) do is to predict the mask vector based solely on the hint, using no information in $\hat{\mathbf{x}}$, but exploiting correlations between $\mathbf{m}$ and $\mathbf{h}$. □.

1.3. Proof of Proposition 1

We prove the first claim by constructing an example in which multiple solutions to (12) exist. Suppose that $\mathbf{H}$ is independent of $\mathbf{M}$ (note that this is equivalent to simply not having a hinting mechanism, since we assume $\mathbf{H}$ is (conditionally) independent of $\mathbf{X}$ given $\mathbf{M}$).

Then (12) becomes

$$\hat{p}(\mathbf{x}|m_i = t) = \hat{p}(\mathbf{x}), \text{ for all } i \in \{1, ..., d\}, t \in \{0, 1\}. \tag{2}$$

We then note that (2) holds if and only if

$$\hat{p}(\mathbf{x}|m_i = 0) = \hat{p}(\mathbf{x}|m_i = 1), \text{ for all } i \in \{1, ..., d\}. \tag{3}$$

Consider the case when $\mathbf{X} = (X_1, X_2, X_3)$ with each X_i being a Bernoulli random variable. Then the number of parameters to be specified by the generator (and therefore the number of parameters in p and $\hat{p}$) is 38, whereas the total number of linear equalities defined by (3) is 24. Therefore $\hat{p}$ is not uniquely determined. □.

1.4. Proof of Lemma 2

The result follows immediately by observing that $p_m(m_i = t|h_i = t) = 1$ for $t \in \{0, 1\}$ and by Lemma 1. □.

1.5. Proof of Proposition 2

Let $\mathbf{m}_0, \mathbf{m}_1 \in \{0, 1\}^d$ be such that they differ only on one component and let this component be the ith with the ith component of $\mathbf{m}_0$ being 0 and of $\mathbf{m}_1$ being 1. Then the hint defined by

$$h_j = \begin{cases} m_j \text{ if } j \neq i \\ 0.5 \text{ if } j = i \end{cases} \tag{4}$$

is such that $p_h(\mathbf{h}|m_i = t) > 0$ for $t = 0$ *and* $t = 1$. By Theorem 1 we then have that

$$\hat{p}(\mathbf{x}|\mathbf{h}, m_i = 0) = \hat{p}(\mathbf{x}|\mathbf{h}, m_i = 1) \tag{5}$$

for all $\mathbf{x} \in \mathcal{X}$.

But then note

$$\hat{p}(\mathbf{x}|\mathbf{h}, m_i = t) = \hat{p}(\mathbf{x}|\mathbf{m}_t, \mathbf{b}) = \frac{\hat{p}(\mathbf{x}|\mathbf{m}_t)}{\mathbb{P}(\mathbf{B} = \mathbf{b}|\mathbf{M} = \mathbf{m}_t)} = \frac{\hat{p}(\mathbf{x}|\mathbf{m}_t)}{\mathbb{P}(\mathbf{B} = \mathbf{b})} \tag{6}$$

where the final equality follows from independence of $\mathbf{B}$ and $\mathbf{M}$.

Substituting this into (5) and multiplying by $\mathbb{P}(\mathbf{B} = \mathbf{b})$ we get

$$\hat{p}(\mathbf{x}|\mathbf{m}_0) = \hat{p}(\mathbf{x}|\mathbf{m}_1). \tag{7}$$

Note that this holds for any $\mathbf{m}_0$, $\mathbf{m}_1$ that differ in at most one component.

Now let $\mathbf{m}_1, \mathbf{m}_2$ be *any* two vectors in $\{0, 1\}^d$. Then there exists $k \in \mathbb{N}$ and a sequence of vectors $\mathbf{m}'_1, ..., \mathbf{m}'_k$ such that $\mathbf{m}'_i$ and $\mathbf{m}'_{i+1}$ differ only on the ith component and $\mathbf{m}_1 = \mathbf{m}'_1$ and $\mathbf{m}_2 = \mathbf{m}'_k$.

It follows from (7) that

$$\hat{p}(\mathbf{x}|\mathbf{m}_1) = \hat{p}(\mathbf{x}|\mathbf{m}'_1) = ... = \hat{p}(\mathbf{x}|\mathbf{m}'_k) = \hat{p}(\mathbf{x}|\mathbf{m}_2) \tag{8}$$

and so in particular, for any $\mathbf{m} \in \{0, 1\}^d$ we have that

$$\hat{p}(\mathbf{x}|\mathbf{m}) = \hat{p}(\mathbf{x}|\mathbf{1}). \tag{9}$$

Since $\hat{p}(\mathbf{x}|\mathbf{1})$ is the density of $\mathbf{X}$, it is unique, and so, the density satisfying (12) is necessarily unique. □.

2. Dataset description in details

2.1. Synthetic data generation

We construct a synthetic dataset using a joint Gaussian distribution with zero means and randomly generated co-variance matrix (C). More specifically, each element of a matrix A is randomly sampled from the uniform distribution $\mathcal{U}(0, 1/2)$. Then, the co-variance matrix C is given by $C = A + A^T$.

Table 1. Statistics of the datasets. S_{cont}: the number of continuous variables, S_{cat}: the number of categorical variables, Avg. Corr: the average absolute correlations among features.

Dataset	N	$\mathbf{S_{cont}}$	$\mathbf{S_{cat}}$	Avg. Corr
Breast	569	30	0	0.3949
Spam	4,601	57	0	0.0608
Letter	20,000	0	16	0.1829
Credit	30,000	14	9	0.1633
News	39,797	44	14	0.0688

3. Hyper-parameters

In all experiments, the depth of the generator and discriminator in both GAIN and auto-encoder is set to 3. The number of hidden nodes in each layer is d, $d/2$ and d, respectively. We use $\tanh$ as the activation functions of each layer except for the output layer where we use the sigmoid activation function and the number of batches is 64 for both the generator and discriminator. For the GAIN algorithm, we use cross-validation to select α among $\{0.1, 0.5, 1, 2, 10\}$.

We use tensorflow to implement GAIN and auto-encoder. We use R to implement MICE (R package MICE (Buuren & Groothuis-Oudshoorn, 2011)), MissForest (R package MissForest (Stekhoven, 2013)), Matrix Completion (R package softImpute (Hastie & Mazumder, 2015)), and EM (R package Amelia (Honaker et al., 2011)).

4. Additional experiments

4.1. Understanding GAIN

Fig. 1 visualizes the convergence of the generator and the discriminator on synthetic data generated as described above. The top figure shows the ground truth of the mask matrix. The three left figures show the estimations of the mask matrix made by the discriminator. As the epoch number increases, it becomes more difficult for the discriminator to distinguish the imputed and observed components (the generator produces imputed components closer to the ground truth). The three right figures show the imputation accuracy of the generator. As the epoch number increases, the imputed components are closer to the ground truth of the missing components.

4.2. Learning curves of GAIN

Fig 2 illustrates the learning curves of the generator and the discriminator of GAIN. As seen in Fig 2 (a), the cross entropy loss of the discriminator converges around $0.7 (\simeq \log(0.5))$ which represents that the discriminator is barely able to distinguish the observed and imputed components. As seen in Fig 2 (b), the RMSE loss of the generator converges to less than 0.1 indicating that the generator imputes the missing components accurately. This result is consistent with Fig. 1.

4.3. Categorical value imputation

As an additional performance metric for comparing the imputation quality of categorical features, we compute the proportion of falsely classified entries (PFC) defined by

$$PFC = \frac{\sum_{i=1}^{n} \sum_{j \in \mathcal{S}_{cat}} (1 - m_j(i)) \mathbb{I}(\hat{x}_j(i) \neq x_j(i))}{\sum_{i=1}^{n} \sum_{j \in \mathcal{S}_{cat}} (1 - m_j(i))},$$

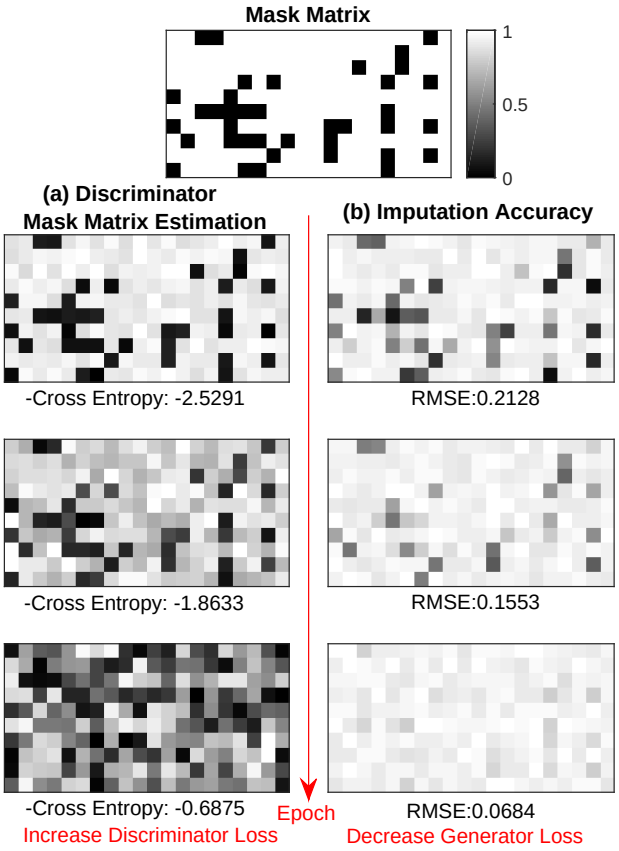


Figure 1. Visualization of the convergence of GAIN. (a) Discriminator outputs, (b) Imputation accuracy of the generator

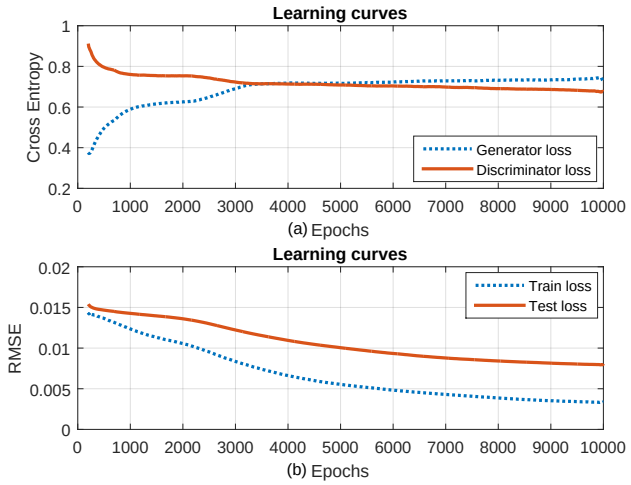


Figure 2. Learning curves of GAIN. (a) Cross entropy loss, (b) RMSE loss

where $\mathcal{S}_{cat}$ is the set of categorical variables. Note that a lower PFC indicates better imputation. We evaluate the PFC performance on the 3 real datasets which have categorical variables (Letter, Credit, News). As can be seen in Table 2, GAIN outperforms all the benchmarks in all the datasets in terms of the PFC metric as well.

Table 2. Imputation performance in terms of PFC (Average ± Std)

Algorithms	Letter	Credit	News
GAIN	**.714 ± .002**	**.478 ± .011**	**.063 ± .002**
MICE	.756 ± .001	.491 ± .013	.072 ± .008
MissForest	.787 ± .001	.499 ± .015	.068 ± .002
Matrix	.779 ± .002	.533 ± .029	.142 ± .003
Auto-encoder	.748 ± .003	.575 ± .016	.065 ± .009
EM	.801 ± .002	.738 ± .010	.091 ± .002

4.4. Estimation of joint distribution

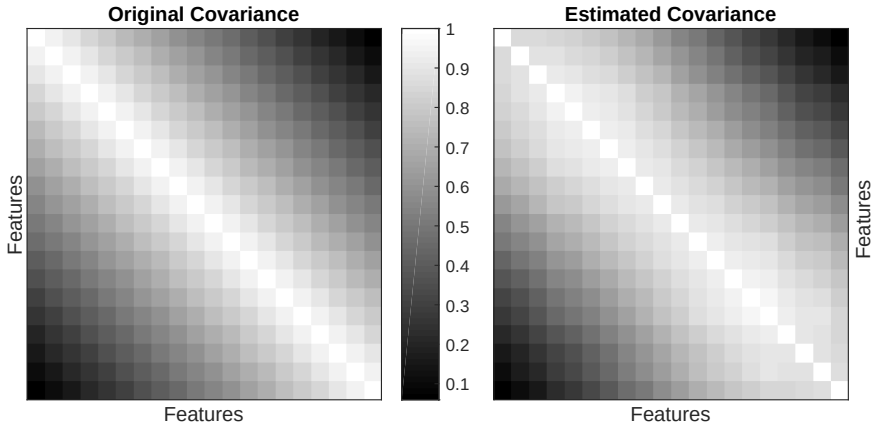


Figure 3. The estimation of joint distribution using GAIN. (a) Original co-variance, (b) Estimated co-variance by GAIN

Fig. 3 shows the qualitative performance of GAIN in estimating the joint distribution $P(\mathbf{X})$. In this experiment, we again use synthetic data and randomly remove 50% of the data. Then, we use GAIN to impute the missing components and estimate the co-variance matrix. As verified by Fig. 3 (a) and (b), the covariance matrix estimated from the GAIN-generated dataset is close to the covariance matrix of the actual data. This verifies that the distribution of $\hat{\mathbf{X}}$ is similar to that of $\mathbf{X}$.

4.5. Preliminary results in error concealment with MNIST

Fig. 4 and 5 visualize the performance of GAIN using MNIST data (LeCun & Cortes, 2010) as an application to error concealment. In this experiment, we randomly remove 50% of the original MNIST image and use GAIN to impute those missing components. Fig 4 shows the imputed images by GAIN after 100, 500, 1000, and 5000 epochs. As the epochs increase, the imputed images become more like the original image.

Fig. 5 shows a visualization of performing multiple imputations (Rubin, 2004) using GAIN. Three imputed images are generated with differently sampled random values $\mathbf{z} \sim \mathcal{U}((0,1)^d)$. The three imputed images are different which qualitatively shows that GAIN captures the uncertainty in the missing components.

4.6. Types of Missingness

As mentioned in Section 1, there are three types of missingness. Here we recall the definition of the first, and formalize the other two, and then provide empirical results demonstrating GAIN's performance in each of these settings.

MCAR: Data is said to be Missing Completely at Random (MCAR) if:

$$\mathbf{X} \perp\!\!\!\perp \mathbf{M} \tag{10}$$

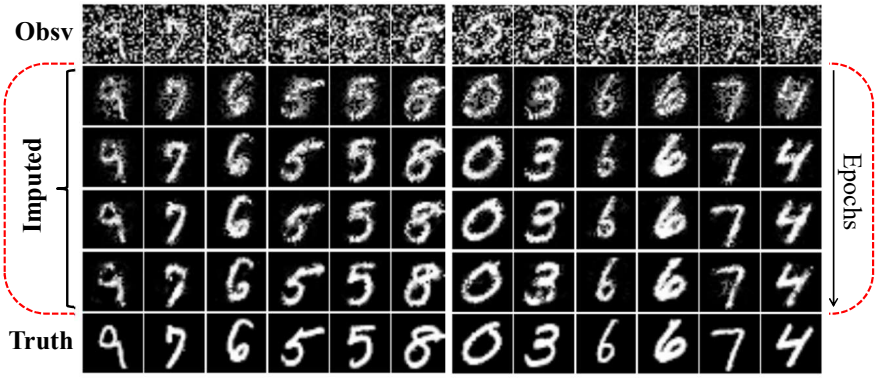


Figure 4. The results of error concealment applied to MNIST dataset. The first row represents the original image with 50% missing. From the second to fifth rows represent the imputed image with different epochs (100, 500, 1000, 5000) by GAIN. The last row represents the original image.

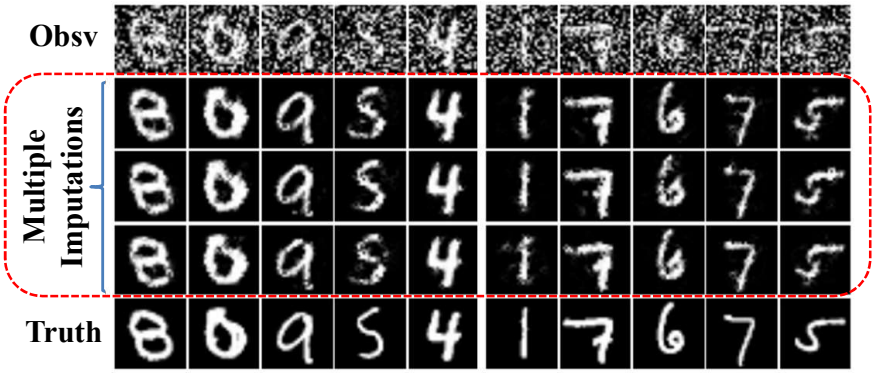


Figure 5. Multiple imputation results with MNIST dataset. Rows 2-4 represent the multiple imputed images by GAIN.

MAR: Data is said to be Missing at Random (MAR) if:

$$\begin{aligned} \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}, \mathbf{m} \in \{0,1\}^d \text{ s.t. } \tilde{\mathbf{x}}_1 = \tilde{\mathbf{x}}_2 \text{ (w.r.t. } \mathbf{m}) \\ \mathbb{P}(\mathbf{M} = \mathbf{m} | \mathbf{X} = \mathbf{x}_1) = \mathbb{P}(\mathbf{M} = \mathbf{m} | \mathbf{X} = \mathbf{x}_2) \end{aligned} \tag{11}$$

MNAR: Data is said to be Missing Not at Random (MNAR) if it is neither MCAR or MAR (in particular, the missingness can depend on the values of the *unobserved* data points).

The following explains how we constructed datasets that satisfy the latter two mechanisms.

Missing at random (MAR): To create an MAR dataset, we sequentially define the probability that the ith component of the nth sample is observed conditional on the missingness and values (if observed) of the previous $i-1$ components to be

$$P_i^m(n) = \frac{p^m(i) \cdot N \cdot e^{-\sum_{j<i} w_j m_j(n) x_j(n) + b_j(1-m_j(n))}}{\sum_{l=1}^N e^{-\sum_{j<i} w_j m_j(l) x_j(l) + b_j(1-m_j(l))}}$$

where $p^m(i)$ corresponds to the average missing rate of the ith feature, and w_j, b_j are sampled from $\mathcal{U}(0,1)$ (but are only sampled once for the entire dataset). We sequentially sample $m_1, ..., m_d$ for each feature vector.

Missing not at random (MNAR): To create an MNAR dataset, we define the probability that the ith component of the nth sample is observed ($P_i^m(n)$) to be

$$P_i^m(n) = \frac{p^m(i) \cdot N \cdot e^{-w_i x_i(n)}}{\sum_{l=1}^N e^{-w_i x_i(l)}}$$

where again $p^m(i)$ corresponds to the average missing rate of the ith feature and w_i is sampled from $\mathcal{U}(0, 1)$. In particular, the missingness of a data point is directly dependent on its value (with dependence determined by the weight w_i).

We compare the RMSE of GAIN against other imputation algorithms on both an MAR and MNAR version of the Credit dataset. To make a fair comparison, we pass the mask matrix to all the benchmarks as an additional input so that they can also utilize the informative missingness captured by it.

Different missing rates for different features: In order to also explore the effect of different missing rates across features on the imputation performance of GAIN, we compare the MCAR, MAR and MNAR settings when $p^m(i) = 0.2 \; \forall i \in \{1, ..., d\}$ (uniform) and when $p^m(i) = 0.4 \times \hat{p}^m(i)$ where $\hat{p}^m(i) \sim \mathcal{U}(0, 1)$ (non-uniform). The average missing rate in both cases is 0.2.

Table 3. Imputation performance with uniform and non-uniform $p^m(i)$ in terms of MCAR, MAR, and MNAR (Average $\pm$ Std of RMSE)

Setting	**Uniform**			**Non-uniform**		
	MCAR	**MAR**	**MNAR**	**MCAR**	**MAR**	**MNAR**
GAIN	**.1858 $\pm$ .0010**	**.1974 $\pm$ .0006**	**.4046 $\pm$ .0053**	**.2114 $\pm$ .0007**	**.2245 $\pm$ .0008**	**.4672 $\pm$ .0066**
MICE	.2585 $\pm$ .0011	.2574 $\pm$ .0035	.5310 $\pm$ .0207	.2574 $\pm$ .0014	.2344 $\pm$ .0068	.5355 $\pm$ .0036
MissForest	.1976 $\pm$ .0015	.2194 $\pm$ .0065	.4286 $\pm$ .0087	.2496 $\pm$ .0065	.2537 $\pm$ .0097	.4784 $\pm$ .0102
Matrix	.2602 $\pm$ .0073	.2473 $\pm$ .0070	.4328 $\pm$ .0036	.2356 $\pm$ .0022	.2440 $\pm$ .0122	.5216 $\pm$ .0084
Auto-encoder	.2388 $\pm$ .0005	.2405 $\pm$ .0070	.4876 $\pm$ .0097	.2444 $\pm$ .0037	.2498 $\pm$ .0129	.5017 $\pm$ .0078
EM	.2604 $\pm$ .0015	.2755 $\pm$ .0063	.5157 $\pm$ .0039	.2620 $\pm$ .0010	.3339 $\pm$ .0024	.4998 $\pm$ .0053

As can be seen in Table 3, GAIN outperforms other state-of-the-art imputation methods in all three missingness settings (both when feature missingness is uniform and non-uniform) and shows significantly better performance in the MNAR setting.

As can also be seen from the right hand side of Table 3, GAIN still outperforms all benchmarks in the non-uniform setting, although the performance of both GAIN and MissForest (its closest competitor in the uniform setting) both decrease similarly, while MICE and Matrix completion both show improvements for the non-uniform setting.

Note that the standard deviation of the total number of missing points is higher for non-uniform $p^m(i)$ than uniform $p^m(i)$. As consistent with Fig. 2(a), higher/lower missing rates yield higher/lower imputation errors; and so, due to the increased standard deviation, there is a greater variance in the performance in the non-uniform setting.

References

Buuren, S. and Groothuis-Oudshoorn, K. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45(3), 2011.

Hastie, T. and Mazumder, R. *softImpute: Matrix Completion via Iterative Soft-Thresholded SVD*, 2015. URL https://CRAN.R-project.org/package=softImpute. R package version 1.4.

Honaker, J., King, G., and Blackwell, M. Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7): 1–47, 2011. URL http://www.jstatsoft.org/v45/i07/.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Rubin, D. B. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

Stekhoven, D. J. *missForest: Nonparametric Missing Value Imputation using Random Forest*, 2013. R package version 1.4.

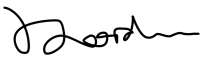
Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	GAIN: Missing Data Imputation using Generative Adversarial Nets
Publication Status	☑ Published □ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Jinsung Yoon, James Jordon, and Mihaela van der Schaar. GAIN: Missing Data Imputation using Generative Adversarial Nets. In *International Conference on Machine Learning*, 2018.

Student Confirmation

Student Name:	James Jordon		
Contribution to the Paper	• I wrote/re-wrote most of the paper, in particular writing the problem formulation, methodology and theoretical analysis sections. • I conducted all of the theoretical analysis for the paper. • I proposed passing noise of equal dimension to missing inputs rather than full-dimensional. • I jointly analysed the experiments with Jinsung and proposed the source of gain experiment. • All authors contributed to the development of the paper through multiple discussions. • All authors reviewed the final draft.		
Signature 		Date	09/06/2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Professor Mihaela van der Schaar		
Supervisor comments I agree that the candidate has made a substantial contribution to the publication.		
Signature 	Date	09/06/2021

This completed form should be included in the thesis, at the end of the relevant chapter.

Chapter 4

Feature Selection

Publications Included

- **James Jordon**, Jinsung Yoon and Mihaela van der Schaar. KnockoffGAN: Generating Knockoffs for Feature Selection using Generative Adversarial Networks. In *International Conference on Learning Representations*, 2019. Weblink: `https://openreview.net/pdf?id=ByeZ5jC5YQ`. [Accepted]

KNOCKOFFGAN: GENERATING KNOCKOFFS FOR FEATURE SELECTION USING GENERATIVE ADVERSARIAL NETWORKS

James Jordon
Engineering Science Department
University of Oxford, UK
james.jordon@wolfson.ox.ac.uk

Jinsung Yoon
Department of Electrical and Computer Engineering
UCLA, California, USA
jsyoon0823@g.ucla.edu

Mihaela van der Schaar
University of Cambridge, UK
Department of Electrical and Computer Engineering, UCLA, California, USA
Alan Turing Institute, London, UK
mihaela@ee.ucla.edu

ABSTRACT

Feature selection is a pervasive problem. The discovery of relevant features can be as important for performing a particular task (such as to avoid overfitting in prediction) as it can be for understanding the underlying processes governing the true label (such as discovering relevant genetic factors for a disease). Machine learning driven feature selection can enable discovery from large, high-dimensional, non-linear observational datasets by creating a subset of features for experts to focus on. In order to use expert time most efficiently, we need a principled methodology capable of controlling the False Discovery Rate. In this work, we build on the promising Knockoff framework by developing a flexible knockoff generation model. We adapt the Generative Adversarial Networks framework to allow us to generate knockoffs with *no assumptions on the feature distribution*. Our model consists of 4 networks, a generator, a discriminator, a stability network and a power network. We demonstrate the capability of our model to perform feature selection, showing that it performs as well as the originally proposed knockoff generation model in the Gaussian setting and that it outperforms the original model in non-Gaussian settings, including on a real-world dataset.

1 INTRODUCTION

Feature selection is a pervasive problem. Often the goal is to discover features that are relevant to a particular outcome, either for the sake of discovery itself or to aid in prediction [16; 25]. When the focus is on discovery, feature selection methods typically focus on trying to control either the Family-Wise Error Rate (FWER) or the False Discovery Rate (FDR). The FWER measures the probability of making a single false discovery (a Type I error) among the selected features (i.e. selecting one which is not relevant), whereas the FDR measures the proportion of false discoveries made (i.e. the proportion of selected features which are false). Controlling FWER, however, leads to reduced power (i.e. selecting fewer relevant variables) since it controls the probability of making *any* false discovery, whereas FDR tries to control the proportion of false discoveries.

Controlling the FDR is important [5; 6; 3]. Often, data-driven feature selection will be used to select a set of candidate features for further investigation. When further investigation is expensive (for example when further investigation would involve conducting new experiments and collecting more data), a method that cannot control the FDR may result in a large amount of wasted resources, with no guarantee that anything meaningful will be discovered. On the other hand, being able to control the FDR at, say, 10% ensures that at most, 10% of the spent resources are wasted, and 90% are in fact spent on discovering positive, useful results. It should be noted, however, that estimating

the FDR of a method empirically is hard in practice, since we do not have access to the ground truth relevance. As such, a theoretical analysis of the method and its (potential) FDR-controlling properties must be carried out, which does not exist for many existing feature selection methods.

[3] is the seminal paper on the knockoff framework, which is an innovative FDR-controlling feature selection method. Knockoffs are features that are generated to "look like" the real features but be conditionally independent of the label given the real features. Feature statistics (such as the coefficients of a LASSO [32]) are compared between the real features and their knockoffs and a selection is made when this difference is sufficiently large. Performing the selection in this way allows for an estimate of the FDR to be obtained and the selection threshold can be adjusted to control the FDR at the selected level. In the original paper, the relationship between the label and the features is constrained to be of a very specific form; in [7], they remove this constraint and instead provide a theoretical analysis that shifts the burden of knowledge onto knowing the underlying feature distribution. Unfortunately, while the theoretical results hold for any feature distribution, they rely on being able to generate valid knockoffs, for which [7] only provide a method for generating knockoffs when the distribution is a (known) multivariate Gaussian distribution. In this paper, we modify the Generative Adversarial Networks (GAN) [11] framework to address this problem, allowing us to generate knockoffs for any distribution (and without any prior knowledge of it). GANs have been shown to be a powerful method for learning to generate complex distributions [24; 20; 2].

Our main contribution is in modifying the discriminator used in the GAN framework in such a way that the generator learns to generate knockoffs satisfying the necessary swap condition [7] which requires that when a feature and its knockoff are swapped, the joint distribution remains unchanged. In addition, we propose a method for maximizing the power of our model using Mutual Information Neural Estimation (MINE) [4] and investigate a regularization method to improve the stability of training. Our model consists of four networks: (1) a generator network that takes as input noise and the real features, and outputs a set of candidate knockoff features; (2) a discriminator network taking as input "swapped" feature-knockoff features that attempts to determine which variables have been swapped; (3) a Wasserstein GAN discriminator that we use as a regularization term; and (4) a MINE network that estimates the mutual information between each feature-knockoff pair allowing us to maximize the power of the knockoff procedure.

2 Related Works

Feature selection is a well-studied problem with a wealth of related works ([12; 31; 17; 22] provide a summary of a lot of existing literature); however, most methods do not attempt to control the FDR. The most common feature selection method for FDR control is the Benjamini-Hochberg (BHq) procedure and its variants [5; 6], which relies on obtaining valid marginal p-values for each selection.

Knockoffs are an active area of research [9; 18; 10]. The notion of a knockoff was first introduced in [3] with the theory there requiring that the relationship between the features and the label be of a specific form. In [7], they build on the knockoff framework, removing this requirement but instead shifting the requirement to one of knowing the distribution of the features. As noted in the introduction, the theory in [7] holds independent of the distribution of the features - relying only on being able to generate valid knockoffs (which exist for any distribution of features). However, they only propose a method for generating knockoffs when the distribution of features is jointly Gaussian. While they do propose a method for generating approximate knockoffs in the non-Gaussian setting (by simply approximating the features as Gaussian), the guarantees on FDR control do not hold for their approximate knockoffs. In [26] and [10], they add to the class of constructible knockoffs, describing methods for constructing knockoffs for Markov Chains, Hidden Markov Models and Gaussian Mixture Models. Though once again, knowledge of the full distribution is still necessary for their construction.

In this paper we use a framework motivated by GANs [11] to learn to generate knockoffs without *any* assumptions on the distribution of the features. To do this, we modify the discriminator so that rather than trying to determine whether a sample is real or fake, it attempts to identify which components have been "swapped". In [38], an unconventional discriminator is used that performs component-wise discrimination for the purpose of imputation. While the problem addressed in that paper is different to the one here, the key idea relies on a similar modification to the discriminator to be able to appropriately guide the generator.

In order to maximize the power of our variable selection mechanism, it will be desirable that the feature-knockoff pairs are "as independent as possible" (this is discussed in [7]). In order to achieve this we will investigate the use of a promising recent paper, MINE [4]. MINE proposes a neural architecture and training procedure capable of estimating the mutual information between two random variables. As the mutual information between two random variables is zero only when they are independent, we will use this as a measure of independence and attempt to minimize it during the training of our modified GAN.

3 BACKGROUND

In this section we introduce our notation and define knockoffs as in [7]. Let us denote the feature space by $\mathcal{X}$ and the label space by $\mathcal{Y}$. Let the dimension of $\mathcal{X}$ be d. Suppose that $\mathbf{X} = (X_1, ..., X_d)$ and Y are random variables over $\mathcal{X}$ and $\mathcal{Y}$. As in [7], we will work with the notion of a *null set*.

Definition 1. *A variable X_j is said to be "null" if and only if Y is independent of X_j conditional on $\{X_i : i \neq j\}$. We define $\mathcal{H}_0$ to be the set of all null variables.*

Our goal will be to discover as many relevant features as possible while controlling the FDR. For a given (potentially random) selection procedure that selects $\hat{\mathcal{S}} \subset \{1, ..., d\}$, we define the FDR to be

$$\text{FDR} = \mathbb{E}\left[\frac{|\hat{\mathcal{S}} \cap \mathcal{H}_0|}{|\hat{\mathcal{S}}|}\right].$$

Note that this agrees with the usual notion of FDR (i.e. when defined in terms of the Markov blanket) under mild assumptions (for a more thorough discussion see [7]).

3.1 KNOCKOFFS

Definition 2. *A knockoff [7] for $\mathbf{X}$ is a random variable $\tilde{\mathbf{X}} \in \mathcal{X}$ satisfying the following two properties:*

$$(\mathbf{X}, \tilde{\mathbf{X}}) \overset{d.}{=} (\mathbf{X}, \tilde{\mathbf{X}})_{swap(S)} \tag{1}$$

$$\tilde{\mathbf{X}} \perp\!\!\!\perp Y | \mathbf{X} \tag{2}$$

for all $S \subset \{1, ..., d\}$ where $(\cdot, \cdot)_{swap(S)}$ denotes the vector obtained by swapping the ith component with the $(i + d)$th component for each $i \in S$ and $\overset{d.}{=}$ is equality in distribution.

In order to use knockoffs for feature selection, we must define an appropriate feature statistic, W_j, that depends on $\mathbf{X}, \tilde{\mathbf{X}}$ and Y, i.e. $W_j = w_j((\mathbf{X}, \tilde{\mathbf{X}}), Y)$ for some function w_j. This function w_j must satisfy the following flip-sign property:

$$w_j((\mathbf{X}, \tilde{\mathbf{X}})_{\text{swap}(S)}, Y) = \begin{cases} w_j((\mathbf{X}, \tilde{\mathbf{X}}), Y) \text{ if } j \notin S \\ -w_j((\mathbf{X}, \tilde{\mathbf{X}}), Y) \text{ if } j \in S. \end{cases} \tag{3}$$

One of the procedures used in [7] to construct these statistics is to perform LASSO, treating the augmented feature-knockoffs as the features on which to regress. This gives LASSO coefficients $b_1, ..., b_{2d}$, and the statistic W_j is set to be the LASSO Coefficient Difference given by

$$W_j = |b_j| - |b_{j+d}|.$$

Note that the FDR control guarantees hold independently of the choice of statistic, but a poorly chosen statistic can significantly impact the power of the test. In particular, using the LASSO Coefficient Difference in non-linear settings can yield few discoveries. The focus of this paper, however, is on generating the knockoffs, not on the statistic used on top of the generated knockoffs and so in our synthetic experiments, we use a linear model for Y to be able to draw fair comparisons between our model and [7]. In the real world data experiment, we use a statistic based on Random Forests for both methods [37].

The following result from [7] depends only on having obtained knockoffs that satisfy Definition 2 and feature statistics satisfying (3) (and in particular do not depend specifically on using LASSO to obtain the statistics).

Theorem 1. *Let $q \in [0,1]$. Given test statistics, $W_1, ..., W_d$, satisfying (3), let*

$$\tau = \min\left\{t > 0 : \frac{1 + |\{j : W_j \leq -t\}|}{|\{j : W_j \geq t\}|} \leq q\right\}.$$

Then the procedure selecting the variables

$$\hat{\mathcal{S}} = \{j : W_j \geq \tau\}$$

controls the FDR at level q, i.e.

$$\mathbb{E}\left[\frac{|\hat{\mathcal{S}} \cap \mathcal{H}_0|}{|\hat{\mathcal{S}}| \vee 1}\right] \leq q.$$

4 KnockoffGAN

It should be noted that in order to satisfy equation (2), it simply needs to be the case that the knockoffs are constructed without looking at the label, Y. In order to satisfy equation (1) we use a modified GAN framework, which gives us the flexibility to learn to generate knockoffs without any assumptions on the distribution of the original features.

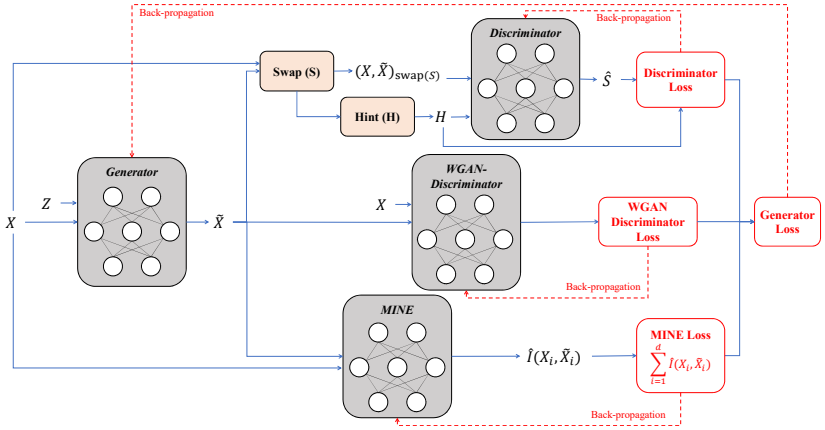


Figure 1: KnockoffGAN Block Diagram

4.1 Generator

The generator, G, will be a function $G(\cdot, \cdot; \phi) : \mathcal{X} \times [0,1]^c \to \mathcal{X}$, parametrized by ϕ that takes a realization $\mathbf{x}$ of $\mathbf{X}$ and random noise, $\mathbf{z} \sim \mathcal{U}([0,1]^c)$, as inputs and outputs knockoff features $\tilde{\mathbf{x}}$. We define $\tilde{\mathbf{X}} := G(\mathbf{X}, \mathbf{z})$. We model G as a fully connected neural network with weights ϕ.

4.2 Discriminator

The main innovation of our paper is in defining the discriminator. Equation (1) imposes a condition on the joint distribution of $(\mathbf{X}, \tilde{\mathbf{X}})$ and as such we must define a discriminator with a loss that is (not necessarily uniquely) minimized only for joint distributions satisfying this condition. To that end, the discriminator, D, will be a function $D(\cdot; \psi) : \mathcal{X} \times \mathcal{X} \to [0,1]^d$ that takes as input a *swapped* sample-knockoff pair $(\mathbf{x}, \tilde{\mathbf{x}})_{\text{swap}(S)}$ and outputs a vector in $[0,1]^d$ with the ith component of $D((\mathbf{x}, \tilde{\mathbf{x}})_{\text{swap}(S)})$ corresponding to the probability that $i \in S$. The discriminator is attempting to detect which variables have been swapped and, intuitively, when the discriminator is unable to determine this, the swapped and unswapped joint distributions must be the same.

The loss we use to train the discriminator is the multi-output cross-entropy loss given by

$$\mathcal{L}_D = \sum_{\mathbf{S} \in \{0,1\}^d} \mathbb{E}_{\mathbf{X} \sim \mathcal{P}_{\mathbf{X}}}[\mathbb{E}_{\tilde{\mathbf{X}} \sim \tilde{\mathcal{P}}_{\mathbf{X}}(\mathbf{X})}[\mathbf{S} \cdot \log(D((\mathbf{X}, \tilde{\mathbf{X}})_{\text{swap}(S)})) + (\mathbf{1} - \mathbf{S}) \cdot \log(1 - D((\mathbf{X}, \tilde{\mathbf{X}})_{\text{swap}(S)}))]] \tag{4}$$

where $\cdot$ is the standard dot, $\mathbf{1} = (1, ..., 1)$, $\mathbf{S} = (S_1, ..., S_d)$ with $S_i = \mathbb{I}(i \in S)$ ($\mathbb{I}$ is the indicator function) and $\log$ is taken element-wise. The following theorem is our main theoretical result, which states that the training regime employed by KnockoffGAN will result in a procedure that generates valid knockoffs.

Theorem 2. *Equation (4) is maximized (with respect to G) if and only if equation (1) is satisfied by G.*

Proof. The proof, alongside supporting theoretical results, can be found in the Appendix. □

In practice, the sum is too computationally expensive ($O(2^d)$) to calculate and so we perform stochastic gradient descent using minibatches with $\mathbf{S}$ sampled uniformly from $\{0, 1\}^d$, independently for each sample in the minibatch.

We also found that training with respect to the full loss resulted in a poor performance, particularly when d is large. We found that the discriminator struggled to learn anything when asked to find the full swap vector, and the poor discriminator resulted in a poorly trained generator. In order to overcome this, we introduce a hint vector - first introduced in [38] - that we use to reveal partial information to the discriminator about the swap vector. We do this by using the hint to reveal some, but not all, of the components of S to the discriminator. In doing so, we reduce the burden of the discriminator from needing to determine the entire swap vector to only needing to determine *some* of the swap vector.

Formally, the hint, $\mathbf{H}$, will be a random variable depending on $\mathbf{S}$, that we pass to the discriminator, alongside $(\mathbf{X}, \tilde{\mathbf{X}})_{\text{swap}(S)}$. We use the hint to control the amount of information we pass to D about $\mathbf{S}$ before asking D to predict $\mathbf{S}$. In practice, our hinting mechanism involves sampling a multivariate Bernoulli random variable, $\mathbf{B}$ from i.i.d. components, which each take value 1 with probability 0.9. The hint is then constructed by setting $H_i = S_i$ if $B_i = 1$ and $H_i = 0.5$ if $B_i = 0$. The discriminator is therefore being asked only to predict the values of $\mathbf{S}$ for which $B_i = 0$; the others, D is able to directly infer from H_i. In order to avoid overfitting to the hint, it becomes necessary to remove these terms from our loss. Our loss now becomes

$$\mathcal{L}_D = \sum_{\mathbf{S} \in \{0,1\}^d} \mathbb{E}_{\mathbf{X} \sim \mathcal{P}_{\mathbf{X}}} \Big[\mathbb{E}_{\tilde{\mathbf{X}} \sim \tilde{\mathcal{P}}_{\mathbf{X}}(\mathbf{X})} \Big[\mathbb{E}_{\mathbf{H} \sim \mathcal{P}_{\mathbf{H}|\mathbf{S}}} \big[(\mathbf{S} \odot (\mathbf{1} - \mathbf{B})) \cdot \log(D((\mathbf{X}, \tilde{\mathbf{X}})_{\text{swap}(S)}, \mathbf{H})) \\ + ((\mathbf{1} - \mathbf{S}) \odot (\mathbf{1} - \mathbf{B})) \cdot \log(1 - D((\mathbf{X}, \tilde{\mathbf{X}})_{\text{swap}(S)}, \mathbf{H})) \big] \Big] \Big] \tag{5}$$

where $\odot$ denotes element-wise multiplication and the expectation over $\mathbf{B}$ is implicit in the expectation over $\mathbf{H}$.

4.3 Stability

We found that adding a regularization term in the form of a Wasserstein GAN discriminator (with Gaussian Process (GP) regularization) [2], f, aided performance. We note that when equation (1) holds, we must have that $\mathbf{X} \overset{d.}{=} \tilde{\mathbf{X}}$ and so the addition of this regularizing term does not affect the optimal solution to our loss. We model f as a fully connect neural network with weights ν. The loss is given by

$$\mathcal{L}_f = \mathbb{E}\left[f(\mathbf{X}) - f(\tilde{\mathbf{X}}) - \eta(||\nabla_{\hat{\mathbf{X}}} f(\hat{\mathbf{X}})||_2 - 1)^2 \right]$$

where $\epsilon \sim \mathcal{U}[0, 1]$, $\hat{\mathbf{X}} = \epsilon \mathbf{X} + (1 - \epsilon)\tilde{\mathbf{X}}$ and η is a hyper-parameter (set to 10 in practice). Note that we have rewritten the loss to be the negative of the one given in [2], allowing us to write our overall objective as a minimax problem. This loss is added to the generator loss as an additional regularization term.

4.4 Maximizing Power

As noted in [7], it is intuitive that in order to maximize the power of the knockoff selection procedure, we wish to make X_j and $\tilde{X}_j$ as "independent" as possible. Doing so ensures that as little as possible of the dependence between the real feature and the label is present between the knockoff and the label; this allows us to determine whether or not the relationship between the feature and label is only through the feature's correlation with other features, or is in fact a true signal.

In order to achieve maximal independence, we look to minimize the mutual information between each feature and its knockoff. Actually computing the true mutual information requires access to both the joint density of the feature-knockoff pairs and to the marginal densities of each feature and knockoff, which we do not have.

Instead, we look to a promising recent work, Mutual Information Neural Estimation (MINE [4]), that provides a framework for estimating the mutual information using neural networks. To do so, they estimate the mutual information between random variables U and V by performing gradient ascent on the following objective:

$$\sup_{\theta \in \Theta} \mathbb{E}_{\mathbb{P}_{UV}^{(n)}}[T_\theta] - \log(\mathbb{E}_{\mathbb{P}_U^{(n)} \otimes \mathbb{P}_V^{(n)}}[e^{T_\theta}])$$

where $\mathbb{P}_{UV}$ denotes the joint measure of (U, V) with $\mathbb{P}_U = \int_{\mathcal{V}} d\mathbb{P}_{UV}$ and $\mathbb{P}_V = \int_{\mathcal{U}} d\mathbb{P}_{UV}$ denoting the marginal measures. $^{(n)}$ denotes the empircal distribution associated with n i.i.d. samples.

Using MINE we approximate the mutual information between each pair X_j and $\tilde{X}_j$ by using d neural networks[1], $T^1, ..., T^d$, each parametrized by $\theta_1, ..., \theta_d$, that we refer to collectively as the *power network*, and will write P to denote the collection of networks $T^1, ..., T^d$. The mutual information is added using a trade-off parameter λ to the loss for G. Formally, define $\mathcal{L}_P$ by

$$\mathcal{L}_P = \sum_{j=1}^{d} \left(\sum_{i=1}^{n} (T_{\theta_j}^j(x_j^{(i)}, \tilde{x}_j^{(i)})) - \log(\sum_{i=1}^{n} \exp(T_{\theta_j}^j(x_j^{(\kappa(i))}, \tilde{x}_j^{(i)}))) \right)$$

where κ is a random permutation of $[n]$[2] and $^{(i)}$ denotes the ith sample - noting that dependence on G is through $\tilde{\mathbf{X}}$.

4.5 Final Objective

The resulting minimax game played by G, D, W and P is given by

$$\min_G \left(\max_D(\mathcal{L}_D) + \lambda \max_P(\mathcal{L}_P) + \mu \max_f(\mathcal{L}_f) \right)$$

where λ, μ are hyper-parameters (set to 1 in the experiments section).

We train each of G, D, W and P iteratively. Pseudo-code of our knockoff construction algorithm can be found in Algorithm 1 and a visual representation of our architecture in Fig. 1.

After generating knockoffs, feature statistics are computed according to some procedure (in the synthetic experiments we use LASSO and in the real data experiment we use a Random Forest-based statistic [37]). Features are then selected based on these statistics according to Theorem 1.

5 Experiments

In this section we demonstrate the capability of our method to match the results of [7] in settings where their model is correctly specified (i.e. when the underlying feature distribution is Gaussian) and then go on to show, in settings where the underlying feature distribution is non-Gaussian, that our method is able to outperform their Gaussian approximation. We compare to two versions of the BHq method [5; 6] to provide a baseline.

We also perform a qualitative analysis of KnockoffGAN on a real-world dataset. We compare features found by KnockoffGAN to PubMed literature and show that KnockoffGAN discovers several meaningful features for 2 different disease outcomes.

[1] In practice we use a single neural network with diagonalized weights to parallelize these networks.

[2] In our pseudo-code, we write $\mathcal{U}(S_n)$ to denote the uniform distribution over the set of all permutations of $[n] = \{1, ..., n\}$.

Algorithm 1 Pseudo-code of KnockoffGAN

Inputs: mini-batch size $n_{mb} > 0$, **Initialize** parameters $\phi, \psi, \nu, \theta_1, ..., \theta_d$
while Converge **do**
  Discriminator Update
  Sample $\mathbf{x}_1, ..., \mathbf{x}_{n_{mb}}$ from $\mathcal{D}$, $\mathbf{z}_1, ..., \mathbf{z}_{n_{mb}} \sim \mathbb{P}_Z$
  Sample $\mathbf{S}_1, ..., \mathbf{S}_{n_{mb}} \overset{i.i.d}{\sim} \mathcal{U}(\{0,1\}^d)$, $\mathbf{b}_1, ..., \mathbf{b}_{n_{mb}} \sim \text{Ber}(0.9)$
  for $i = 1, ..., n_{mb}$ **do**
    $\tilde{\mathbf{x}}_i \leftarrow G(\mathbf{x}_i, \mathbf{z}_i; \phi)$
    $\mathbf{h}_i = \mathbf{S}_i \odot \mathbf{b}_i + 0.5(\mathbf{1} - \mathbf{b}_i)$
  Update D by ascending its stochastic gradient

$$\nabla_\psi \sum_{i=1}^{n_{mb}} \Big[(\mathbf{S}_i \odot (\mathbf{1} - \mathbf{b}_i)) \cdot \log(D((\mathbf{x}_i, \tilde{\mathbf{x}}_i)_{\text{swap}(\mathbf{S})}), \mathbf{h}_i) + ((\mathbf{1} - \mathbf{S}_i) \odot (\mathbf{1} - \mathbf{b}_i)) \cdot \log(\mathbf{1} - D((\mathbf{x}_i, \tilde{\mathbf{x}}_i)_{\text{swap}(\mathbf{S})}, \mathbf{h}_i)) \Big]$$

  MINE Update
  Sample $\mathbf{x}_1, ..., \mathbf{x}_{n_{mb}}$ from $\mathcal{D}$, $\mathbf{z}_1, ..., \mathbf{z}_{n_{mb}} \sim \mathbb{P}_Z$, $\kappa \sim \mathcal{U}(S_{n_{mb}})$
  for $i = 1, ..., n_{mb}$ **do**
    $\tilde{\mathbf{x}}_i \leftarrow G(\mathbf{x}_i, \mathbf{z}_i; \phi)$
  for $j = 1, ..., d$ **do**
    Update T_j by ascending its stochastic gradient

$$\nabla_{\theta_j} \Big(\textstyle\sum_{i=1}^{n_{mb}} T^j_{\theta_j}(x^{(i)}_j, \tilde{x}^{(i)}_j) \Big) - \log \Big(\textstyle\sum_{i=1}^{n_{mb}} \exp(T^j_{\theta_j}(x^{(i)}_j, \tilde{x}^{(\kappa(i))}_j)) \Big)$$

  WGAN-GP Update
  Sample $\mathbf{x}_1, ..., \mathbf{x}_{n_{mb}}$ from $\mathcal{D}$, $\mathbf{z}_1, ..., \mathbf{z}_{n_{mb}} \sim \mathbb{P}_Z$
  for $i = 1, ..., n_{mb}$ **do**
    Sample $\epsilon \sim \mathcal{U}[0, 1]$
    $\tilde{\mathbf{x}}_i \leftarrow G(\mathbf{x}_i, \mathbf{z}_i; \phi)$
    $\hat{\mathbf{x}}_i = \epsilon \mathbf{x}_i + (1 - \epsilon)\tilde{\mathbf{x}}_i$
  Update f by ascending its stochastic gradient

$$\nabla_\nu \textstyle\sum_{i=1}^{n_{mb}} \Big[f(\mathbf{x}_i) - f(\tilde{\mathbf{x}}_i) - \eta(||\nabla_{\hat{\mathbf{x}}_i} f(\hat{\mathbf{x}}_i)||_2 - 1)^2 \Big]$$

  Generator Update
  Sample $\mathbf{x}_1, ..., \mathbf{x}_{n_{mb}}$ from $\mathcal{D}$, $\mathbf{z}_1, ..., \mathbf{z}_{n_{mb}} \sim \mathbb{P}_Z$
  Sample $\mathbf{S}_1, ..., \mathbf{S}_{n_{mb}} \overset{i.i.d}{\sim} \mathcal{U}(\{0,1\}^d)$, $\kappa \sim \mathcal{U}(S_{n_{mb}})$
  for $i = 1, ..., n_{mb}$ **do**
    $\tilde{\mathbf{x}}_i \leftarrow G(\mathbf{x}_i, \mathbf{z}_i; \phi)$
  Update G by descending its stochastic gradient

$$\nabla_\phi(\mathcal{L}_D + \lambda \mathcal{L}_P + \mu \mathcal{L}_f)$$

5.1 Synthetic Data Experiments

5.1.1 Simulation settings

Evaluating feature selection methods on real data is difficult as we do not have access to the ground truth. To evaluate KnockoffGAN, we conduct a series of experiments using *synthetic data*, replicating those carried out in [7] and extending them to more general settings. In each of the following synthetic experiments, we set the feature dimension to be $d = 1000$ and the number of samples to be $n = 3000$. For each feature distribution we perform two experiments:

1. Y-Logit: $P(Y = 1|\mathbf{X}) = \frac{\exp(m(\mathbf{X}))}{(1+\exp(m(\mathbf{X})))}$
2. Y-Gaussian: $Y \sim \mathcal{N}(m(\mathbf{X}), 1)$

where $m(\mathbf{X}) = \sum_{i=1}^{60} \alpha \delta_i X_i$ with $\delta_i \in \{-1, 1\}$ sampled uniformly and then fixed for each experiment. α controls the strength of the influence that $\mathbf{X}$ has on Y, and in the experiments we vary this (as in [7]). Note that for the auto-regressive settings (found in Section 5.1.2 and the Appendix) the relevant variables are sampled uniformly at random from among the 1000 features (rather than being the first 60); in the non-auto-regressive settings this is not necessary.

We report the True Positive Rate (TPR), which is also commonly referred to as the *power*, defined as

$$\text{TPR} = \frac{|\hat{\mathcal{S}} \cap \mathcal{S}^*|}{|\mathcal{S}^*|} \tag{6}$$

where $\mathcal{S}^* = \{1, ..., d\} \setminus \mathcal{H}_0$ is the set of all non-null features. We also report the FDR to verify that the methods do indeed control it at the specified level which we set to be 10%. Note that we are not using FDR as a metric - a lower FDR is not desirable when we set the threshold to 10%. In fact, we want the methods to be as close to 10% as possible (so that they are achieving maximum power). We perform 100 replications of each experiment and report the average TPR and FDR.

5.1.2 Gaussian settings

We begin by replicating the setup from [7] in which the underlying feature distribution is Gaussian. In this setting, we do not expect KnockoffGAN to perform better than the original knockoff framework as the original framework assumes a Gaussian distribution. Our goal here is simply to achieve a similar performance, demonstrating that little performance is lost even when the distribution is known to be Gaussian.

In the first experiment that we replicate from [7], the features are set to be auto-regressive (AR(1)) with Gaussian marginal distributions, i.e. $X_i = \phi X_{i-1} + Z_i$ with Z_i being chosen such that $X_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \frac{1}{n})$. In this experiment we vary ϕ, which determines the correlation between features, rather than α. We fix $\alpha = 3.5$ for Y-Gaussian and $\alpha = 10$ for Y-Logit. The results are reported in Fig. 2.

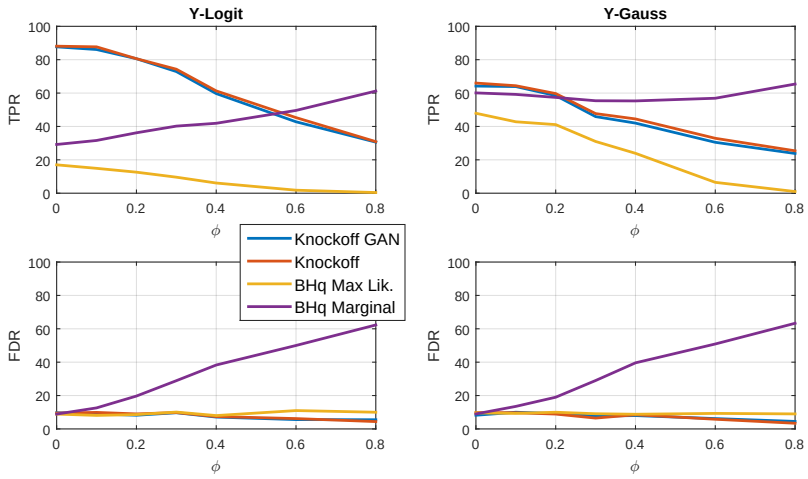


Figure 2: Comparison of KnockoffGAN with the benchmarks for $\mathbf{X}$ distributed as an auto-regressive distribution with Gaussian marginal distributions. TPR is used to quantify performance and FDR is reported to verify that it is at the specified threshold (10%).

As in [7], we observe that BHq Marginal, which tests for marginal independence of the feature from Y, suffers from severely increased FDR as we increase the correlation, invalidating the seemingly good TPR. To make the remaining results clearer, we omit BHq marginal from the rest of this section. Aside from this, we see in Fig. 2, that the other methods control the FDR at or very close to the specified 10% threshold. We also see that across the entire range of α, KnockoffGAN achieves a very similar TPR to the original Knockoff framework.

In the second experiment, we set the underlying feature distribution to be i.i.d. Gaussian. We found in this case also that KnockoffGAN was able to control the FDR and achieve a similar TPR to the

original knockoff framework. More details of this experiment and the results for it can be found in the Appendix

5.1.3 NON-GAUSSIAN SETTINGS

We now move on to the key results for the paper in which the underlying feature distribution is no longer Gaussian. In this setting, we expect to outperform the original Knockoff framework due to the fact that they approximate the distribution as Gaussian. In particular, when this approximation is poor, the knockoffs are no longer valid and as such no FDR guarantees can be given. On the other hand, KnockoffGAN does not place any requirements on the distribution of the features and as such is able to generate valid knockoffs.

We performed experiments for several different underlying feature distributions, and found that KnockoffGAN achieved a higher TPR than the original knockoff framework in all cases, while controlling the FDR at the specified level. We give the results for $\mathbf{X}$ coming from a 4-Gaussian mixture model in Fig. 3 - results for Uniform, Dirichlet, and other (2 and 3) Gaussian mixture models can be found in the Appendix.

To create our 4-mixture model, we set the means $(\mathbf{m}^1, \mathbf{m}^2, \mathbf{m}^3, \mathbf{m}^4)$ of the 4 Gaussians to be:

- $m_i^1 = 1$ for $i = 1$ to 100 and 0 for $i = 101$ to 1000,
- $m_i^2 = 1$ for $i = 1$ to 50 and -1 for $i = 51$ to 100 and 0 for $i = 101$ to 1000,
- $m_i^3 = -1$ for $i = 1$ to 50 and 1 for $i = 51$ to 100 and 0 for $i = 101$ to 1000,
- $m_i^4 = -1$ for $i = 1$ to 100 and 0 for $i = 101$ to 1000.

We scale the variance of each Gaussian to be such that the overall variance of each feature is $\frac{1}{n}$.

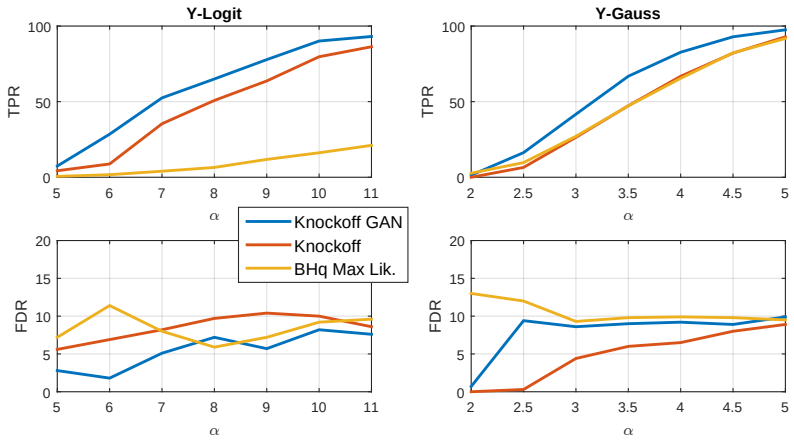


Figure 3: Comparison of KnockoffGAN with the benchmarks for $\mathbf{X}$ distributed as a 4-mixture Gaussian mixture model. TPR is used to quantify performance and FDR is reported to verify that it is at the specified threshold (10%).

We see in Fig. 3 that KnockoffGAN consistently outperforms the original knockoff framework, achieving a higher TPR across the entire range of α while consistently controlling the FDR at 10%. In fact, in the Y-Gaussian setting we see that the original knockoff framework performs almost identically to BHq Maximum Likelihood.

5.1.4 IMPACT OF WGAN REGULARIZATION

We conclude the synthetic experiments by demonstrating the effect of the WGAN regularizer[3]. We conduct this experiment using an auto-regressive model with $\mathcal{U}(-\sqrt{3/n}, \sqrt{3/n})$ marginal distributions. We fix $\alpha = 5$ for Y-Logit and $\alpha = 2.5$ for Y-Gauss.

[3]The WGAN regularizer was included in all previous experiments.

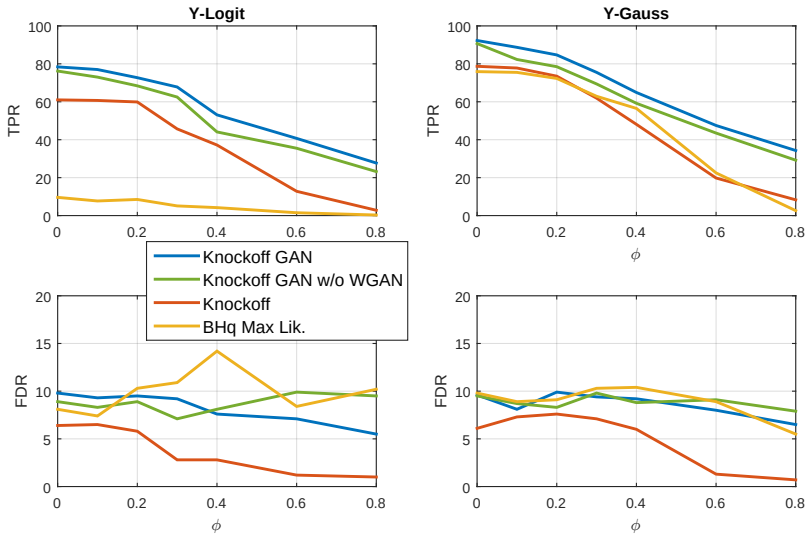


Figure 4: A comparison of the performance of KnockoffGAN with and without the WGAN regularizer for $\mathbf{X}$ distributed as an auto-regressive distribution with $\mathcal{U}(-\sqrt{3/n}, \sqrt{3/n})$ marginal distributions. TPR is used to quantify performance and FDR is reported to verify that it is at the specified threshold (10%).

As we see in Fig. 4, the WGAN regularizer has a significant effect on the results, with the improvement in some places being almost as much as KnockoffGAN without WGAN makes over the original knockoff framework. As noted in Section 4.3, there is no trade-off introduced by the inclusion of this regularizer; the optimal solution to the loss is unchanged and therefore this regularization is "free" in terms of FDR control, but as demonstrated improves TPR performance.

5.2 Real Data Experiment

In this section we use a biobank dataset (with 387 dimensions) to qualitatively analyze the performance of KnockoffGAN. We use KnockoffGAN to select features for two different outcomes: (1) Cardiovascular Disease (CVD) and (2) Diabetes and then use PubMed literature to asses the validity of the selected features.

We found that the original knockoff framework was unable to select even the most well-known features (such as Age and Sex for CVD [15]), even when the FDR threshold was increased to 20%. Therefore, there are no relevant features to report for the original knockoff framework and so Table 1 contains only the features selected by KnockoffGAN that were deemed relevant by PubMed literature. For this the FDR threshold was set to 5% so that the number of discoveries was manageable for cross-reference with PubMed.

No	Cardiovascular Disease	PubMed ref.	Diabetes	PubMed ref.
1	Age	[15]	Lipid-lowering drugs	[35]
2	Sex	[15]	Comparative body size	[27]
3	Daily smoking	[1]	Home owned	[13]
4	FEV1	[28]	Insomnia	[34]
5	Diastolic blood pressure	[33]	Anti-hypertensive drugs	[8]
6	Diabetes	[29]	Asthma	[30]
7	Father chronic bronchitis	[14; 23]	Height	[19; 27]
8	Alcohol intake	[21]	Alcohol intake	[36]
9	Long-standing illness			

Table 1: Discovered features using KnockoffGAN framework, verified using PubMed literature. The FDR threshold was set to 5%.

As we see in Table 1, KnockoffGAN discovers 9 relevant features for CVD and 8 relevant features for diabetes. Some of the relevant features, such as Age, Sex and Long-standing illness for CVD are

well-known relevant features. The remaining features are all supported by the literature in PubMed. While this is a qualitative result (it relies on using PubMed as the ground truth), we do believe this demonstrates that KnockoffGAN is a significant improvement over the original knockoff generation procedure.

6 CONCLUSION

In this paper we built on the knockoff framework introduced in [3] by developing a novel GAN framework, KnockoffGAN, capable of generating knockoffs with no assumptions on the underlying data. We demonstrated through a series of experiments on a range of synthetic datasets and on a real world dataset that our method improves on the performance of the original knockoff framework.

While we feel this is a significant step towards being able to generate knockoffs for any data, there is still more work to be done. In particular, generalizing this method to time-series data would be non-trivial, and would be an interesting avenue for further investigation.

ACKNOWLEDGEMENT

The authors would like to thank the reviewers for their helpful comments. The research presented in this paper was supported by the Office of Naval Research (ONR) and the NSF (Grant number: ECCS1462245, ECCS1533983, and ECCS1407712).

REFERENCES

[1] John A Ambrose and Rajat S Barua. The pathophysiology of cigarette smoking and cardiovascular disease: an update. *Journal of the American college of cardiology*, 43(10):1731–1737, 2004.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] Rina Foygel Barber and Emmanuel J Candes. A knockoff filter for high-dimensional selective inference. *arXiv preprint arXiv:1602.03574*, 2016.

[4] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Devon Hjelm, and Aaron Courville. Mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 530–539, 2018.

[5] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pp. 289–300, 1995.

[6] Yoav Benjamini and Daniel Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, pp. 1165–1188, 2001.

[7] Emmanuel Candès, Yingying Fan, Lucas Janson, and Jinchi Lv. Panning for gold: Model-free knockoffs for high-dimensional controlled variable selection. *arXiv preprint arXiv:1610.02351*, 2016.

[8] Antonio Ceriello, Dario Giugliano, Antonio Quatraro, and Pierre J Lefebvre. Anti-oxidants show an anti-hypertensive effect in diabetic and hypertensive subjects. *Clinical Science*, 81(6): 739–742, 1991.

[9] Yingying Fan, Emre Demirkaya, Gaorong Li, and Jinchi Lv. Rank: large-scale inference with graphical nonlinear knockoffs. *arXiv preprint arXiv:1709.00092*, 2017.

[10] Jaime Roquero Gimenez, Amirata Ghorbani, and James Zou. Knockoffs for the mass: new feature importance statistics with false discovery guarantees. *arXiv preprint arXiv:1807.06214*, 2018.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[12] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[13] C Jaffiol, F Thomas, K Bean, B Jégo, and N Danchin. Impact of socioeconomic status on diabetes and cardiovascular risk factors: results of a large french survey. *Diabetes & metabolism*, 39(1):56–62, 2013.

[14] Pekka Jousilahti, Erkki Vartiainen, Jaakko Tuomilehto, and Pekka Puska. Symptoms of chronic bronchitis and the risk of coronary disease. *The Lancet*, 348(9027):567–572, 1996.

[15] Pekka Jousilahti, Erkki Vartiainen, Jaakko Tuomilehto, and Pekka Puska. Sex, age, cardiovascular risk factors, and coronary heart disease: a prospective follow-up study of 14 786 middle-aged men and women in finland. *Circulation*, 99(9):1165–1172, 1999.

[16] Alan Jović, Karla Brkić, and Nikola Bogunović. A review of feature selection methods with applications. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on*, pp. 1200–1205. IEEE, 2015.

[17] Alan Jović, Karla Brkić, and Nikola Bogunović. A review of feature selection methods with applications. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on*, pp. 1200–1205. IEEE, 2015.

[18] Eugene Katsevich and Chiara Sabatti. Multilayer knockoff filter: Controlled variable selection at multiple resolutions. *arXiv preprint arXiv:1706.09375*, 2017.

[19] D Lawlor, S Ebrahim, and G Davey Smith. The association between components of adult height and type ii diabetes and insulin resistance: British women's heart and health study. *Diabetologia*, 45(8):1097–1106, 2002.

[20] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, pp. 4, 2017.

[21] Michael Marmot and Eric Brunner. Alcohol and cardiovascular disease: the status of the u shaped curve. *BMJ: British Medical Journal*, 303(6802):565, 1991.

[22] Tahir Mehmood, Kristian Hovde Liland, Lars Snipen, and Solve Sæbø. A review of variable selection methods in partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 118:62–69, 2012.

[23] Howraman Meteran, Vibeke Backer, Kirsten Ohm Kyvik, Axel Skytthe, and Simon Francis Thomsen. Heredity of chronic bronchitis: a registry-based twin study. *Respiratory medicine*, 108(9):1321–1326, 2014.

[24] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pp. 271–279, 2016.

[25] Carl Rietschel, Jinsung Yoon, and Mihaela van der Schaar. Feature selection for survival analysis with competing risks using deep learning. *arXiv preprint arXiv:1811.09317*, 2018.

[26] Matteo Sesia, Chiara Sabatti, and Emmanuel J Candès. Gene hunting with knockoffs for hidden markov models. *arXiv preprint arXiv:1706.04677*, 2017.

[27] Suzanne M Shoff and Polly A Newcomb. Diabetes, body size, and risk of endometrial cancer. *American journal of epidemiology*, 148(3):234–240, 1998.

[28] Don D Sin and SF Paul Man. Chronic obstructive pulmonary disease: a novel risk factor for cardiovascular disease. *Canadian journal of physiology and pharmacology*, 83(1):8–13, 2005.

[29] James R Sowers and Melvin A Lester. Diabetes and cardiovascular disease. *Diabetes care*, 22: C14, 1999.

[30] Lars C Stene and Per Nafstad. Relation between occurrence of type 1 diabetes and asthma. *The Lancet*, 357(9256):607–608, 2001.

[31] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. *Data classification: Algorithms and applications*, pp. 37, 2014.

[32] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[33] Ramachandran S Vasan, Martin G Larson, Eric P Leip, Jane C Evans, Christopher J O'donnell, William B Kannel, and Daniel Levy. Impact of high-normal blood pressure on the risk of cardiovascular disease. *New England journal of medicine*, 345(18):1291–1297, 2001.

[34] Alexandros N Vgontzas, Duanping Liao, Slobodanka Pejovic, Susan Calhoun, Maria Karataraki, and Edward O Bixler. Insomnia with objective short sleep duration is associated with type 2 diabetes: a population-based study. *Diabetes care*, 2009.

[35] Sandeep Vijan and Rodney A Hayward. Pharmacologic lipid-lowering therapy in type 2 diabetes mellitus: background paper for the american college of physicians. *Annals of Internal Medicine*, 140(8):650–658, 2004.

[36] SG Wannamethee, AG Shaper, IJ Perry, and KGMM Alberti. Alcohol consumption and the incidence of type ii diabetes. *Journal of Epidemiology & Community Health*, 56(7):542–548, 2002.

[37] Marvin N Wright, Theresa Dankowski, and Andreas Ziegler. Unbiased split variable selection for random survival forests using maximally selected rank statistics. *Statistics in medicine*, 36 (8):1272–1284, 2017.

[38] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. GAIN: Missing data imputation using generative adversarial nets. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5689–5698, 2018.

APPENDIX

THEORETICAL RESULTS

In order to prove Theorem 2, we use similar techniques to those used in the original GAN paper [11]. In what follows, we analyze the minimax game defined by:

$$\min_G \max_D \sum_{\mathbf{S}\in\{0,1\}^d} \mathbb{E}_{\mathbf{X}\sim\mathcal{P}_\mathbf{X}}[\mathbb{E}_{\tilde{\mathbf{X}}\sim\tilde{\mathcal{P}_\mathbf{X}}(\mathbf{X})}[\mathbf{S}\cdot\log(D((\mathbf{X},\tilde{\mathbf{X}})_{\text{swap}(S)}))+(\mathbf{1}-\mathbf{S})\cdot\log(1-D((\mathbf{X},\tilde{\mathbf{X}})_{\text{swap}(S)}))]] \tag{7}$$

where we have used the version of $\mathcal{L}_D$ given by equation (4) in the main manuscript (i.e. without hinting). The theoretical results that follow are proven only for this version of the loss, though we do believe that the theorem holds more generally for the hinting version of the loss - this is backed up by our empirical results demonstrating strict FDR control while using the hint mechanism. After proving that the optimal solution to this game does indeed provide us with valid knockoffs, we will then show that the additional term $\mathcal{L}_f$ does not change the optimal solution. Let p be the density of $(\mathbf{X},\tilde{\mathbf{X}})$.

We begin by stating a lemma, that follows from a similar proof to Proposition 1 in [11].

Lemma 1. *Let $(\mathbf{x},\tilde{\mathbf{x}}) \in \mathcal{X}\times\mathcal{X}$. Then for a fixed generator, G, the i^{th} component of the optimal discriminator, $D^*((\mathbf{x},\tilde{\mathbf{x}}))$ is given by*

$$D^*((\mathbf{x},\tilde{\mathbf{x}}))_i = \frac{p((\mathbf{x},\tilde{\mathbf{x}})_{swap(\{i\})})}{p((\mathbf{x},\tilde{\mathbf{x}})_{swap(\{i\})}) + p((\mathbf{x},\tilde{\mathbf{x}}))} \tag{8}$$

for each $i \in \{1,...,d\}$.

Proof. The proof of this involves some basic integral manipulation to get that the objective in (7) can be rewritten as

$$\sum_{i=1}^{d}\int_{\mathcal{X}\times\mathcal{X}} \log D((\mathbf{x},\tilde{\mathbf{x}}))_i p((\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(\{i\})}) + \log(1-D((\mathbf{x},\tilde{\mathbf{x}}))_i)p((\mathbf{x},\tilde{\mathbf{x}}))\mathrm{d}\mathbf{x}.$$

We then observe that $y \mapsto a\log y + b\log(1-y)$ achieves its maximum in $[0,1]$ at $\frac{a}{a+b}$ and so the objective is maximized (with respect to D, for fixed G) when

$$D^*((\mathbf{x},\tilde{\mathbf{x}}))_i = \frac{p((\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(\{i\})})}{p((\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(\{i\})}) + p((\mathbf{x},\tilde{\mathbf{x}}))}$$

for each $i \in \{1,...,d\}$. □

With this lemma, we are now able to prove our key result.

Theorem 3. *Equation (7) is maximized (with respect to G) if and only if equation (1) (in the main paper) is satisfied by G.*

Proof. We begin by rewriting our loss, substituting in D^*, to give us the following loss for G:

$$\mathcal{L}_G = \sum_{S\subset\{1,\dots,d\}} \mathbb{E}_{\mathbf{X},\tilde{\mathbf{X}}}\Big(\sum_{i\in S}\log\frac{p((\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(S\setminus i)})}{p((\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(S\setminus i)}) + p((\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(S)})} + \sum_{i\notin S}\log\frac{p((\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(S)})}{p((\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(S\cup i)}) + p((\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(S)})}\Big)$$

where we note that

$$((\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(S)})_{\text{swap}(\{i\})} = \begin{cases}(\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(S\setminus i)} \text{ if } i\in S\\ (\mathbf{x},\tilde{\mathbf{x}})_{\text{swap}(S\cup i)} \text{ if } i\notin S.\end{cases}$$

Then by inspecting each term in the sum, we see that each term is a KL-divergence term that is minimized only when

$$p((\mathbf{x}, \tilde{\mathbf{x}})_{\text{swap}(\{S \setminus i\})}) = p((\mathbf{x}, \tilde{\mathbf{x}})_{\text{swap}(S)}) \tag{9}$$

and

$$p((\mathbf{x}, \tilde{\mathbf{x}})_{\text{swap}(\{S \cup i\})}) = p((\mathbf{x}, \tilde{\mathbf{x}})_{\text{swap}(S)})$$

for every $i \in \{1, ..., d\}$, every $S \subset \{1, ..., d\}$ and each $(\mathbf{x}, \tilde{\mathbf{x}}) \in \mathcal{X} \times \mathcal{X}$.

By iteratively applying equation (9), we get that

$$\begin{aligned} p((\mathbf{x}, \tilde{\mathbf{x}})_{\text{swap}(S)}) &= p((\mathbf{x}, \tilde{\mathbf{x}})_{\text{swap}(S \setminus 1)}) = ... = p((\mathbf{x}, \tilde{\mathbf{x}})_{\text{swap}(S \setminus \{1,...,d-1\})}) \\ &= p((\mathbf{x}, \tilde{\mathbf{x}})_{\text{swap}(S \setminus \{1,...,d\})}) = p((\mathbf{x}, \tilde{\mathbf{x}}) \end{aligned}$$

proving the theorem. □

Lemma 2. *The addition of the term $\mathcal{L}_f$ to our loss, does not affect the optimal solution to it.*

Proof. By theorem 2, it suffices to show that any distribution satisfying equation (1) also minimizes $\max_f \mathcal{L}_f$. But we note that, as shown in [2], $\max_f \mathcal{L}_f$ (or rather $\sup_f \mathcal{L}_f$) is the Wasserstein distance between $\mathbf{X}$ and $\tilde{\mathbf{X}}$, which is 0 (and minimal) when $\mathbf{X} \overset{d.}{=} \tilde{\mathbf{X}}$. It therefore suffices to show that equation (1) implies $\mathbf{X} \overset{d.}{=} \tilde{\mathbf{X}}$.

Let $S = \{1, ..., d\}$. Then if $(\mathbf{X}, \tilde{\mathbf{X}})$ satisfy equation (1), we get that $(\mathbf{X}, \tilde{\mathbf{X}}) \overset{d.}{=} (\tilde{\mathbf{X}}, \mathbf{X})$. Since the joint distributions are equal, it follows that the marginal distributions are equal and so by projecting onto the first d variables, we get that $\mathbf{X} \overset{d.}{=} \tilde{\mathbf{X}}$. □

MINE

We state the key theory used by MINE to estimate the mutual information. For full details see the original paper, [4].

The mutual information is defined as

$$I(U;V) = \int_{\mathcal{U}\times\mathcal{V}} \log \frac{d\mathbb{P}_{UV}}{d\mathbb{P}_U \otimes d\mathbb{P}_V} d\mathbb{P}_{UV}$$

where U and V are random variables over some spaces $\mathcal{U}$ and $\mathcal{V}$, respectively with joint measure $\mathbb{P}_{UV}$ and marginal measures $\mathbb{P}_U = \int_{\mathcal{V}} d\mathbb{P}_{UV}$ and $\mathbb{P}_V = \int_{\mathcal{U}} d\mathbb{P}_{UV}$, respectively.

The mutual information can also be characterized by the Kullback-Leibler divergence, D_{KL}, as

$$I(U;V) = D_{KL}(\mathbb{P}_{UV}||\mathbb{P}_U \otimes \mathbb{P}_V)$$

The Donsker-Varadhan representation then gives us for any two probability measures $\mathbb{P}$ and $\mathbb{Q}$ over a probability space Ω

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_{\mathbb{P}}[T] - \log(\mathbb{E}_{\mathbb{Q}}[e^T])$$

where the supremum is taken over all functions T such that the two expectations are finite.

A simple corollary of this is that fixing a class $\mathcal{F}$ of functions (such as a parametrized class $\{T_\theta : \theta \in \Theta\}$) over which the supremum is taken will provide us with a lower bound for the mutual information that approaches the true mutual information as the class becomes sufficiently rich. MINE [4] fix the class to be parametric in this way - given a fixed neural network architecture, they let $\mathcal{F} = \{T_\theta : \theta \in \Theta\}$ be the set of all functions parametrized by this network.

IMPLEMENTATION OF KNOCKOFFGAN

In the experiments, the depth of the generator, discriminator and WGAN-GP networks is set to 4 and power network is set to 3. The number of hidden nodes in each layer is $d/4$, $d/16$ $d/4$ for the generator, discriminator, and WGAN-GP, respectively. For the power network, we use 2 diagonal matrices for each layer to make two hidden nodes for each feature separately. We use ReLu and tanh as the activation functions for each layer except for the output layer where we use a linear activation function for the generator, power network and WGAN-GP networks and sigmoid activation function for the discriminator network. The number of samples in each mini-batch is 128. KnockoffGAN is implemented in tensorflow.

DETAILS OF BENCHMARKS

We use the following links for the implementations of 3 benchmarks.

- Knockoff: `http://web.stanford.edu/group/candes/knockoffs/software/knockoff/index.html`
- BHq Max: `http://web.stanford.edu/group/candes/knockoffs/software/knockoff/tutorial-4-r.html`
- BHq Marginal: Modifying the original code of BHq Max in `http://web.stanford.edu/group/candes/knockoffs/software/knockoff/tutorial-4-r.html`

Except for the knockoff generation step, KnockoffGAN follows the same procedures as the original knockoff framework described at `http://web.stanford.edu/group/candes/knockoffs/software/knockoff/tutorial-2-r.html` to select features.

ADDITIONAL EXPERIMENTS

INDEPENDENT GAUSSIANS

In the following experiment, features were taken to be i.i.d. Gaussian, with mean 0 and variance $\frac{1}{n}$, i.e. $\mathbf{X} \sim \mathcal{N}(0, \frac{1}{n}\mathbf{I}_n)$. The results are reported in Fig. 5.

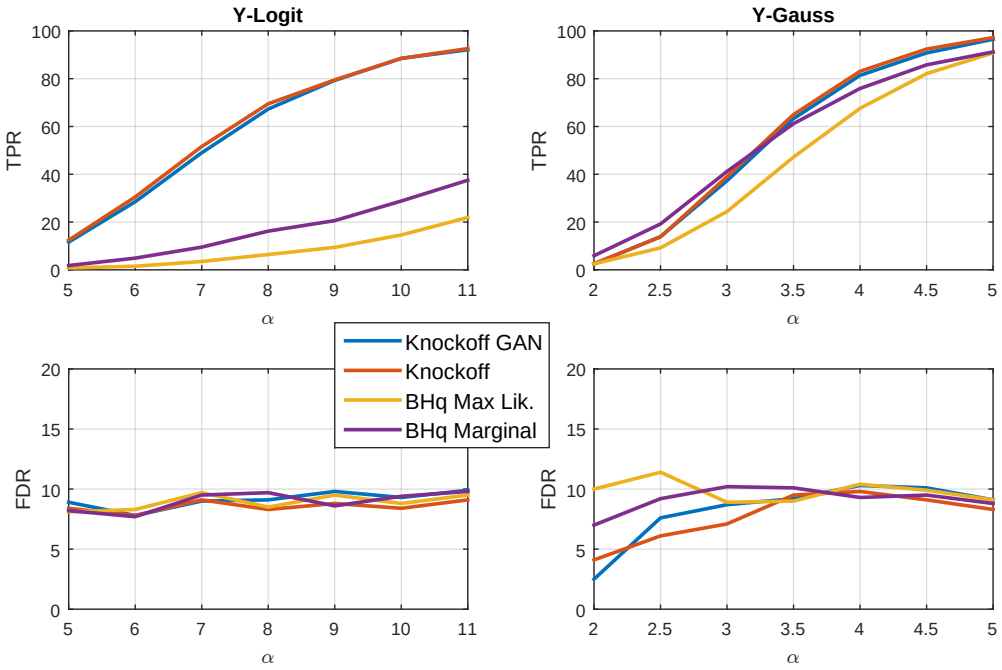


Figure 5: Comparison of KnockoffGAN with the benchmarks for $\mathbf{X} \sim \mathcal{N}(0, \frac{1}{n}\mathbf{I}_n)$. TPR is used to quantify performance and FDR is reported to verify that it is at the specified threshold (10%).

As we see in Fig. 5, all methods control the FDR at or very close to the specified 10% threshold. We also see that across the entire range of α, KnockoffGAN achieves a very similar TPR to the original knockoff framework.

NON-GAUSSIAN SETTINGS

INDEPENDENT UNIFORM

In this experiment we set the feature distribution to be a Uniform distribution with mean 0 and variance $\frac{1}{n}$ (to be consistent with the Gaussian experiments). Fig. 6 displays the results for each component of $\mathbf{X}$ being i.i.d. $\mathcal{U}(-\sqrt{3/n}, \sqrt{3/n})$. Once again we see that KnockoffGAN consistently outperforms the original knockoff framework, achieving a higher TPR across the entire range of α in both settings.

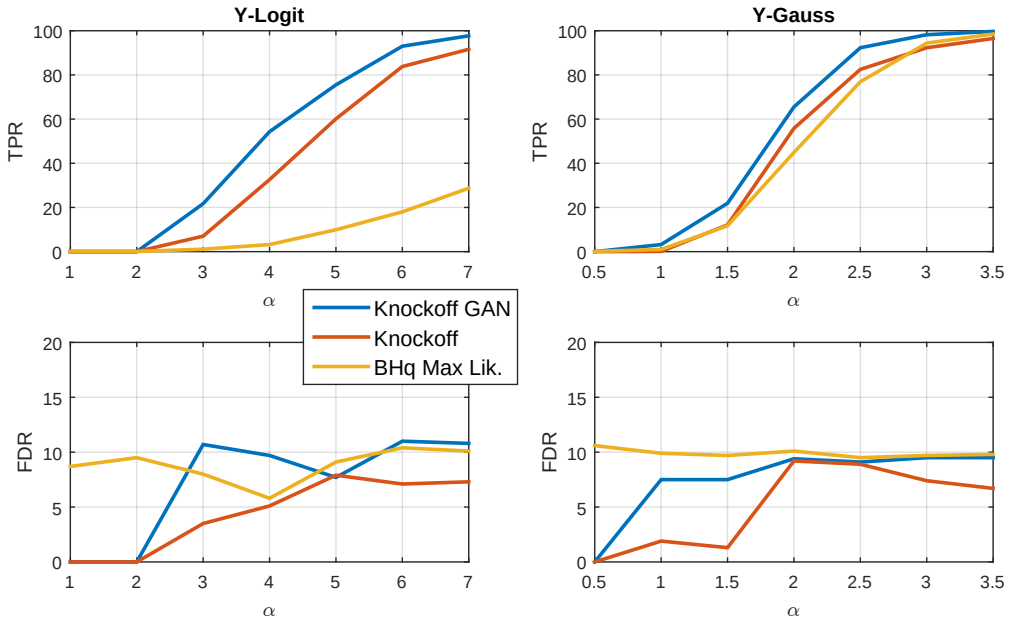


Figure 6: Comparison of KnockoffGAN with the benchmarks for $\mathbf{X} \sim \mathcal{U}(-\sqrt{3/n}, \sqrt{3/n})$. TPR is used to quantify performance and FDR is reported to verify that it is at the specified threshold (10%).

DIRICHLET

In this experiment we set the feature distribution to be a Dirichlet$(1, ..., 1)$ distribution - i.e. the uniform distribution over the $(d-1)$-simplex. Correlation here exists through the requirement that $\sum_{i=1}^{d} X_i = 1$.

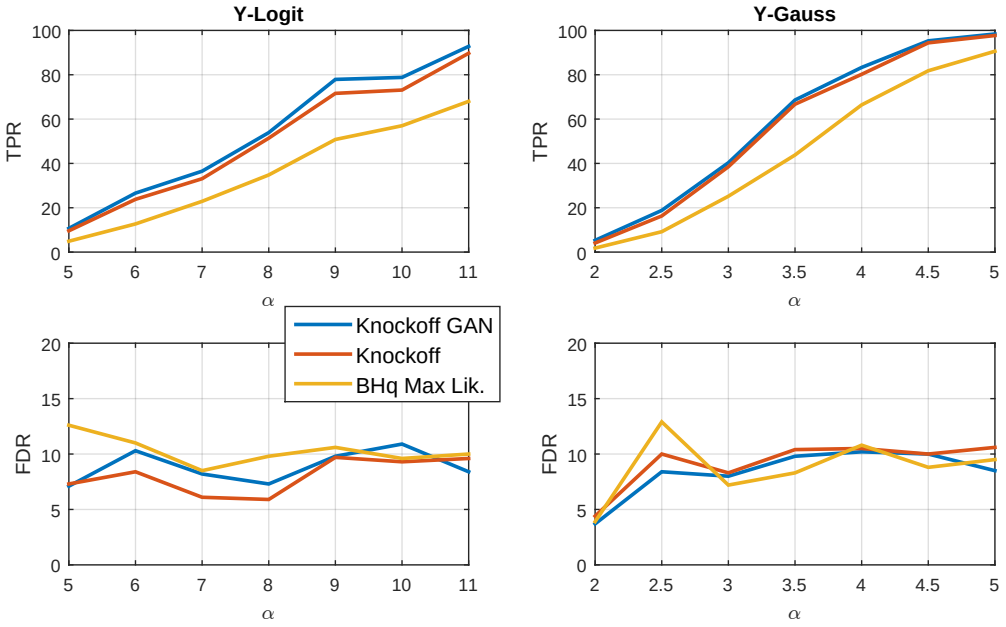


Figure 7: Comparison of KnockoffGAN with the benchmarks for $\mathbf{X} \sim$ Dirichlet$(1, ..., 1)$. TPR is used to quantify performance and FDR is reported to verify that it is at the specified threshold (10%).

GAUSSIAN MIXTURE MODELS

For the GMM2 model we set the means $(\mathbf{m}^1, \mathbf{m}^2)$ of the 2 Gaussians to be:

- $m_i^1 = 1$ for $i = 1$ to 100 and 0 for $i = 101$ to 1000,
- $m_i^2 = -1$ for $i = 1$ to 100 and 0 for $i = 101$ to 1000.

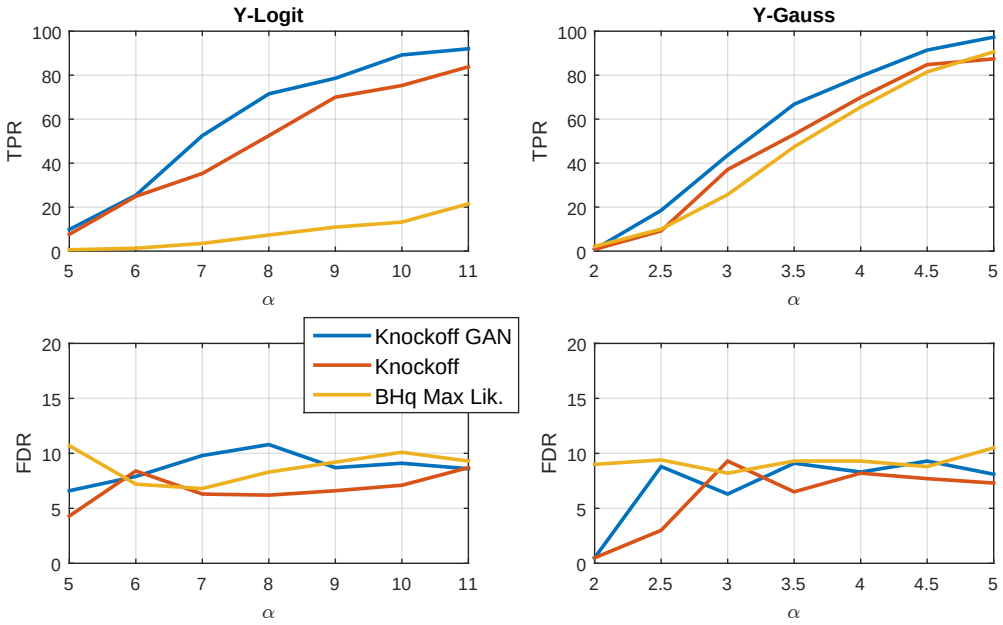


Figure 8: Comparison of KnockoffGAN with the benchmarks for $\mathbf{X} \sim$ GMM2. TPR is used to quantify performance and FDR is reported to verify that it is at the specified threshold (10%).

For the GMM3 model we set the means $(\mathbf{m}^1, \mathbf{m}^2, \mathbf{m}^3)$ of the 3 Gaussians to be:

- $m_i^1 = 1$ for $i = 1$ to 100 and 0 for $i = 101$ to 1000,
- $m_i^2 = 1$ for $i = 1$ to 50 and -1 for $i = 51$ to 100 and 0 for $i = 101$ to 1000,
- $m_i^3 = -1$ for $i = 1$ to 100 and 0 for $i = 101$ to 1000.

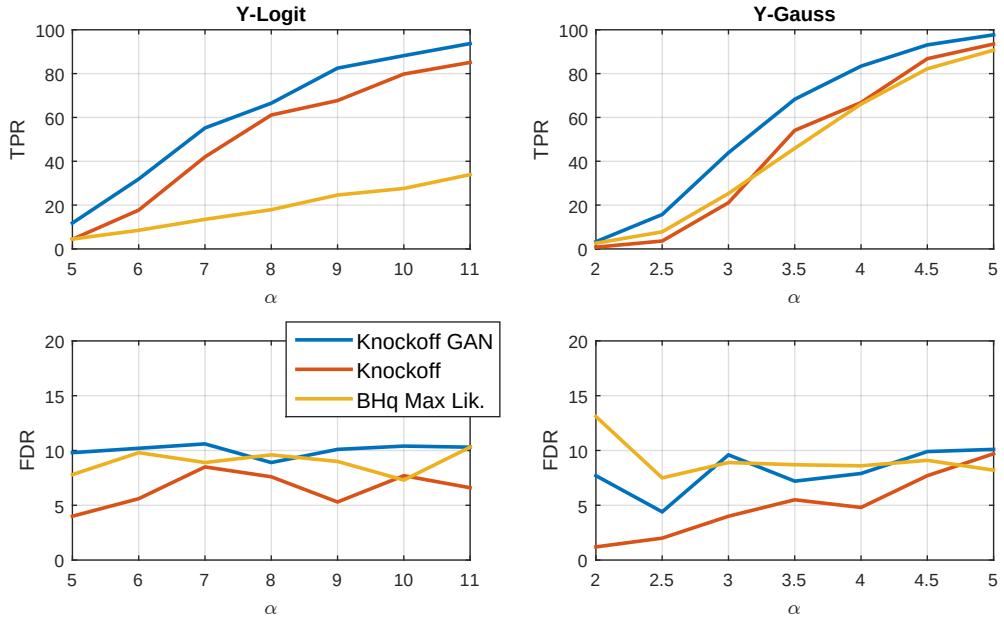


Figure 9: Comparison of KnockoffGAN with the benchmarks for $\mathbf{X} \sim$ GMM3. TPR is used to quantify performance and FDR is reported to verify that it is at the specified threshold (10%).

HYPER-PARAMETER ANALYSIS

Hyper-parameter selection in the feature selection problem is difficult; hyper-parameter selection cannot be performed using cross-validation as we do not have access to ground truth. The hyper-parameters must therefore be fixed a priori. For this we believe that $\lambda = 1$ and $\mu = 1$ are perhaps the most canonical choice we could make and thus these were used in the main manuscript. In the following experiments, we investigate the sensitivity of the results to various settings of λ and μ. The results below are in the auto-regressive Uniform setting with Y-Logit and FDR set to 10%.

TPR		λ					
		0	0.1	0.5	1	5	10
ϕ	0.1	70.3	76.2	75.7	77.0	75.1	76.0
	0.2	68.7	72.6	71.9	72.7	72.9	73.7
	0.4	42.4	55.1	51.1	53.1	53.8	53.2
	0.8	22.4	27.4	25.3	27.7	27.5	27.6

Table 2: Evaluation of the hyper-parameter λ in the auto-regressive uniform setting with Y-Logit and FDR threshold 10%

TPR		μ					
		0	0.1	0.5	1	5	10
ϕ	0.1	70.3	76.2	75.7	77.0	75.1	76.0
	0.2	68.7	72.6	71.9	72.7	72.9	73.7
	0.4	42.4	55.1	51.1	53.1	53.8	53.2
	0.8	22.4	27.4	25.3	27.7	27.5	27.6

Table 3: Evaluation of the hyper-parameter μ in the auto-regressive uniform setting with Y-Logit and FDR threshold 10%

As can be seen in Table 2 and 3, the performance of KnockoffGAN is not sensitive to the value of λ and μ. The only significant difference can be seen when either λ or μ is set to 0 which represents the exclusion of either the power network or WGAN-discriminator network, respectively, from the model. In particular, the lack of sensitivity to μ aligns with Lemma 2 in which we see that there is not a trade-off between $\mathcal{L}_f$ and $\mathcal{L}_D$ but rather the two can be simultaneously minimised.

To further understand the effects of the WGAN network, we report the final values of the other losses ($\mathcal{L}_D$ and $\mathcal{L}_P$) when $\mu = 1$ and $\mu = 0$ (i.e. with and without the WGAN regularisation). The results are given below in the auto-regressive Uniform setting with Y-Logit and FDR set to 10%.

Loss		$\mathcal{L}_D$		$\mathcal{L}_P$	
		$\mu = 0$	$\mu = 1$	$\mu = 0$	$\mu = 1$
ϕ	0.1	0.6894	0.6962	0.0014	0.0203
	0.2	0.7005	0.6964	0.0018	0.0115
	0.4	0.6919	0.6955	0.0046	0.0180
	0.8	0.7013	0.6960	0.0068	0.0311

Table 4: Final values of the other training losses when $\mu = 0$ or $\mu = 1$ in the auto-regressive uniform setting with Y-Logit and FDR threshold 10%

As can be seen in Table 4, the inclusion of the WGAN network improves the final value of $\mathcal{L}_D$, bringing it significantly closer to its optimal value of $\log(2) \approx 0.693$. On the other hand, the power network loss is increased, however, this trade-off is expected and acceptable - the FDR guarantees rely on the discriminator loss ($\mathcal{L}_D$) and not the power network. As we see in Table 3, the TPR does not suffer from this increased loss for the power network.

For our final hyper-parameter evaluation, we investigate the effect of varying the hinting probability, p, which determines the probability with which $B_i = 1$. We note that this hyper-parameter is used to trade-off between speed of learning and optimality of the learned solution. A low probability makes for fast convergence, but suboptimal convergence whereas a high probability makes for slow convergence (but a more optimal solution). We chose 0.9 to balance this, following the implementation of [38]. To demonstrate this trade-off we vary this hyper-parameter (from 0 to 0.9). The results below are for the auto-regressive Uniform distribution with Y-Logit and FDR set to 10%.

TPR		p					
		0	0.2	0.4	0.6	0.8	0.9
ϕ	0.1	73.3	74.7	75.0	75.3	75.7	77.0
	0.2	67.7	68.4	69.1	69.7	72.4	72.7
	0.4	51.2	51.6	51.7	51.9	52.9	53.1
	0.8	18.0	20.4	21.4	22.4	25.4	27.7

Table 5: Evaluation of the hinting probability, p, in the auto-regressive uniform setting with Y-Logit and FDR threshold 10%

RESULTS WITH FDR THRESHOLD 5%

In all of the synthetic experiments above, the FDR threshold is set to 10% (which is the level most thoroughly investigated by [3]). In this final experiment, we set the FDR threshold to 5% (which aligns with our real-world experiment) and verify that KnockoffGAN is capable of controlling the FDR at this level and still outperforms the original knockoff framework. We conduct the experiments on (1) an auto-regressive distribution with Gaussian marginal distributions, (2) an auto-regressive distribution with Uniform marginal distributions, and (3) a 4-mixture Gaussian mixture model.

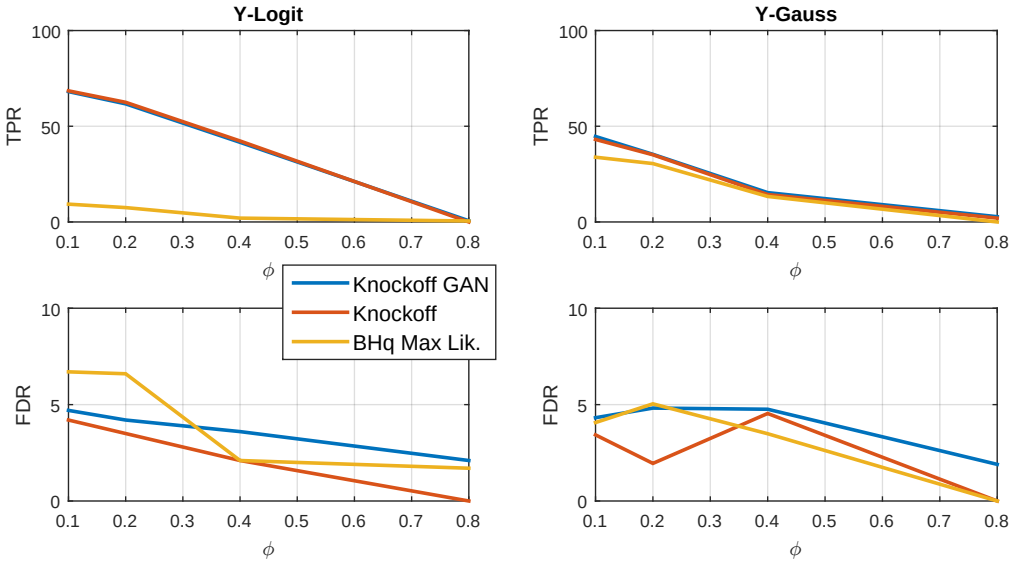


Figure 10: A comparison of the performance of KnockoffGAN for $\mathbf{X}$ distributed as an auto-regressive distribution with Gaussian marginal distributions. TPR is used to quantify performance and FDR is reported to verify that it is at the specified threshold (5%).

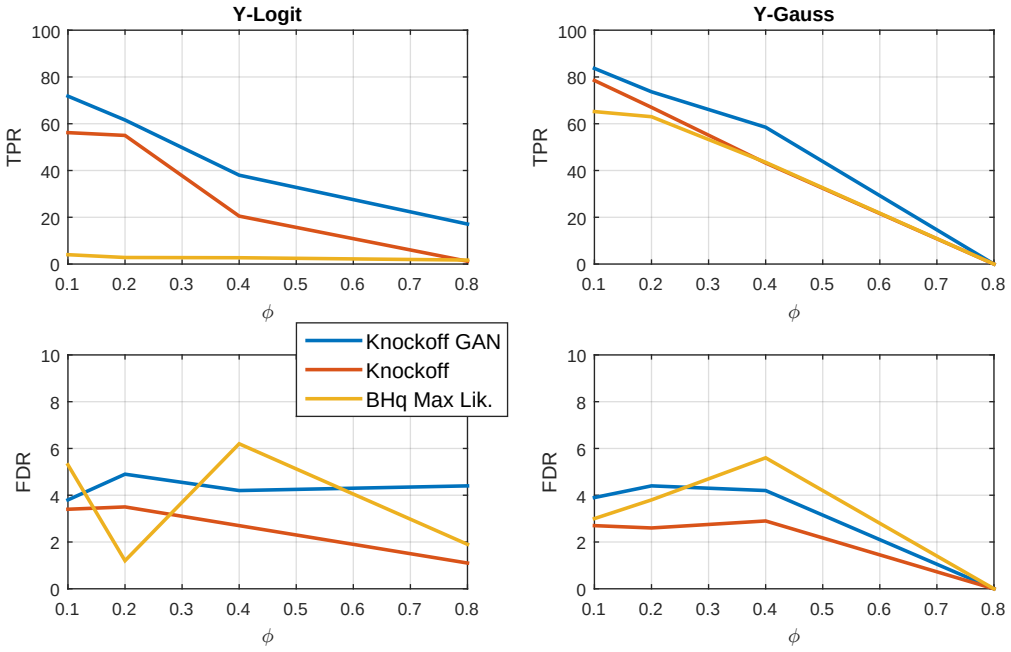


Figure 11: A comparison of the performance of KnockoffGAN for $\mathbf{X}$ distributed as an auto-regressive distribution with Uniform marginal distributions. TPR is used to quantify performance and FDR is reported to verify that it is at the specified threshold (5%).

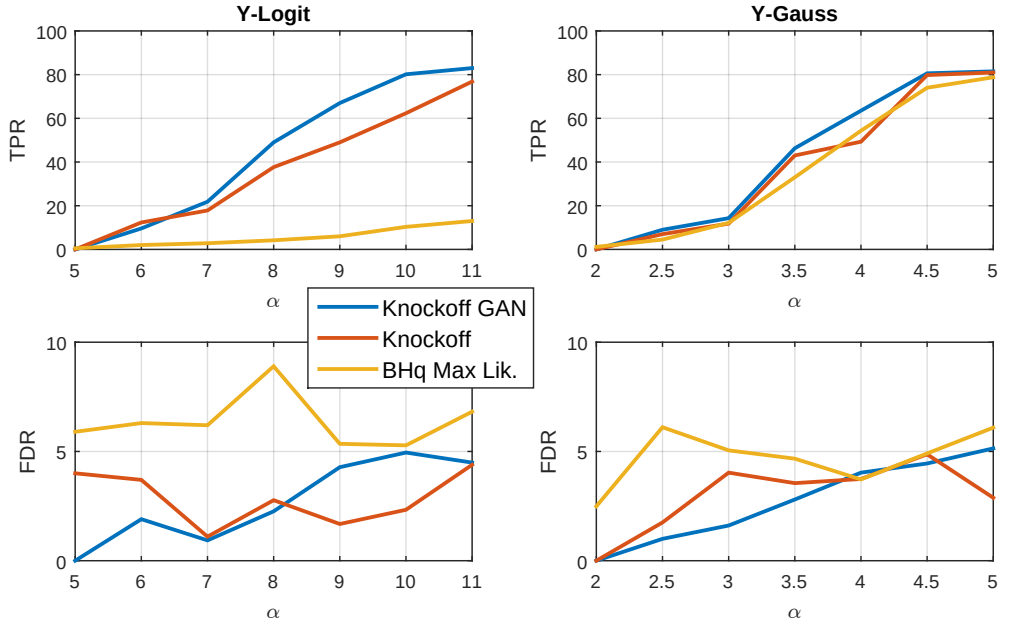


Figure 12: A comparison of the performance of KnockoffGAN for $\mathbf{X}$ distributed as 4-mixture Gaussian mixture model. TPR is used to quantify performance and FDR is reported to verify that it is at the specified threshold (5%).

STATISTICS OF REAL-WORLD BIOBANK DATASET

To preserve anonymity of the authors, the full details of this dataset will be given upon acceptance of the paper. In Table 1, we provide the basic statistics of the real-world biobank dataset.

Staticstics	**Values**
No of patients	**86082**
No of features	**387**
Pearson correlation across the features	25%: **0.0042**, 50%: **0.0120** ,75%: **0.0332**
Pearson correlation with the outcome	**CVD:** 25%: **0.0033**, 50%: **0.0091** ,75%: **0.0199** **Diabetes:** 25%: **0.0068**, 50%: **0.0179** ,75%: **0.0439**
Label distribution	**CVD:** 1252 patients (1.5%) **Diabetes:** 3932 patients (4.6%)

Table 6: Basic statistics of real-world biobank dataset (% means percentile in Pearson correlation rows)

Chapter 5

Private Learning

Publications Included

- **James Jordon***, Jinsung Yoon* and Mihaela van der Schaar. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *International Conference on Learning Representations*, 2019. Weblink: `https://openreview.net/pdf?id=S1zk9iRqF7`. [Accepted]

- **James Jordon**, Jinsung Yoon and Mihaela van der Schaar. Differentially Private Bagging: Improved utility and cheaper privacy than subsample-and-aggregate. In *Advances in Neural Information Processing Systems*, 2020. Weblink: `https://openreview.net/pdf?id=HkljOVBgUH`. [Accepted]

PATE-GAN: GENERATING SYNTHETIC DATA WITH DIFFERENTIAL PRIVACY GUARANTEES

James Jordon*
Engineering Science Department
University of Oxford, UK
james.jordon@wolfson.ox.ac.uk

Jinsung Yoon*
Department of Electrical and Computer Engineering
UCLA, California, USA
jsyoon0823@g.ucla.edu

Mihaela van der Schaar
University of Cambridge, UK
Department of Electrical and Computer Engineering, UCLA, California, USA
Alan Turing Institute, London, UK
mihaela@ee.ucla.edu

ABSTRACT

Machine learning has the potential to assist many communities in using the large datasets that are becoming more and more available. Unfortunately, much of that potential is not being realized because it would require sharing data in a way that compromises privacy. In this paper, we investigate a method for ensuring (differential) privacy of the generator of the Generative Adversarial Nets (GAN) framework. The resulting model can be used for generating synthetic data on which algorithms can be trained and validated, and on which competitions can be conducted, *without* compromising the privacy of the original dataset. Our method modifies the Private Aggregation of Teacher Ensembles (PATE) framework and applies it to GANs. Our modified framework (which we call PATE-GAN) allows us to tightly bound the influence of any individual sample on the model, resulting in tight differential privacy guarantees and thus an improved performance over models with the same guarantees. We also look at measuring the quality of synthetic data from a new angle; we assert that for the synthetic data to be useful for machine learning researchers, the relative performance of two algorithms (trained and tested) on the synthetic dataset should be the same as their relative performance (when trained and tested) on the original dataset. Our experiments, on various datasets, demonstrate that PATE-GAN consistently outperforms the state-of-the-art method with respect to this and other notions of synthetic data quality.

1 INTRODUCTION

More and more large datasets are becoming available in a wide variety of communities. In the U.S. medical community, for example, the fraction of providers using electronic health records (EHR) increased from 9.4% in 2008 to 83.8% in 2015 [20]. The availability of large datasets presents enormous opportunities for collaboration between the data-holders and the machine learning community. However, many of these large datasets, especially EHR, include sensitive information that prevents data-holders from sharing the data.

The most common way to mitigate the privacy risk of sharing sensitive records is to de-identify the records - but it is by now well-known that records that have been de-identified can be easily re-identified by linking them to other identifiable datasets [30; 13; 24; 22; 14]. (This is especially true for medical records of patients who have rare diseases.) However, if the purpose of sharing the data is to develop and validate machine learning methods for a particular task (e.g. prognostic risk scoring), real data is not necessary; it would suffice to have synthetic data that is *sufficiently like* the real data.

*: Equal contribution

Precisely what this means depends on how the synthetic data will be used. For example, the synthetic data may be used to train models that will be deployed directly on real data. In this setting it is important that these methods (which we trained entirely on synthetic data) perform as well as if they had been trained on real data. Another setting to consider is one in which data-holders wish to use the synthetic data to identify the best method(s) to be used on the real data [11]. In this setting, it is not important that training on synthetic data leads to good performance on real data, but rather that comparing two methods on the synthetic data results in conclusions similar to those that would have been drawn from comparing the two methods on the real data. We evaluate our method in both settings.

Generative Adversarial Networks (GAN) [19] provide a powerful method for using real data to generate synthetic data but it does not provide any rigorous privacy guarantees. Our method modifies the GAN machinery in a way that *does* guarantee privacy; the synthetic data is (differentially) private [12] with respect to the original data. To do this we modify the training procedure of the discriminator to be differentially private by using a modified version of the Private Aggregation of Teacher Ensembles (PATE) [25; 26] framework. The Post-Processing Theorem [12] then guarantees that the GAN generator - which is trained only using the differentially private discriminator - will also be differentially private and thus so will the synthetic data it generates. We call our proposed framework PATE-GAN.

Using two Kaggle datasets, two different real-world medical datasets and two UCI datasets, we evaluate the utility of the samples generated by PATE-GAN in various settings with various levels of differential privacy. In line with the settings outlined above, we consider two methods for evaluating the similarity of synthetic datasets with a real dataset. The first method, first proposed in [15], compares the predictive performance of models trained on the synthetic datasets and tested on the real dataset. The second method, which we propose for the first time here, compares the performance rankings of predictive models on the synthetic datasets with their performance rankings on the real dataset. We demonstrate that, for both of these methods, PATE-GAN consistently produces synthetic datasets that are "more like" the original real dataset than the synthetic datasets produced by the state-of-the-art benchmark (DPGAN [32]).

The contributions of this paper can be summarized as follows: (1) we modify the PATE framework and apply it to GANs to generate synthetic data, (2) we demonstrate in the experiments section that using PATE to enforce differential privacy results in higher quality synthetic data than DPGAN using various real-world datasets, (3) we propose a novel new metric for evaluating the generated synthetic data.

2 Related Works

The most related previous work to this paper is DPGAN [32]. Like us, DPGAN proposes a framework for modifying the GAN framework to be differentially private, also relying on the Post-Processing Theorem to change the problem of learning a differentially private generator to learning a differentially private discriminator. Their work uses a technique introduced by [1] that provides a differentially private mechanism for training deep networks. The key idea is that noise is added to the gradient of the discriminator during training to create differential privacy guarantees. These ideas are also used in [2]. Our method is similar in spirit; during training of the discriminator differentially private training data is used, which results in noisy gradients, however, we use the mechanism introduced in [25] which we believe gives tighter differential privacy guarantees (via tighter bounds on the effect of a single sample) than those provided in [1]. This means that for the same privacy guarantees, our method is capable of producing higher quality synthetic data. For a visual representation of both PATE-GAN and DPGAN, see the Appendix.

The proposed model modifies the PATE framework [25; 26] for use in a generative model setting (specifically for use with GANs). The key to the GAN framework is that the discriminator is a *differentiable* module trained to classify samples as either real or generated. The PATE framework provides a differentially private mechanism for classification by training multiple teacher models on disjoint partitions of the data. To classify a new sample each teacher's output is evaluated on the sample and then all outputs are noisily aggregated. This noisy aggregation, though, results in a classifier that is *not* differentiable with respect to the parameters of the generator. In order to overcome this problem we follow the idea of the student model, also proposed in [25], that

involves taking some *public* unlabelled data, labelling it using the standard PATE mechanism and then training the student using the resulting labelled data. Because access to any public data is often an unreasonable assumption in synthetic data generation, we adapt this training paradigm in a way that does not require public data by training the student using only outputs from the (differentially private) generator.

Some previous works generate synthetic data using summary statistics of the original data [23] or based on specific domain-knowledge [6]; however, those methods are limited to low-dimensional feature spaces, specific fields and do not provide any differential privacy guarantees. [9] generates synthetic patient records using a GAN framework. However, [9] focuses only on generating discrete variables, whereas PATE-GAN is capable of generating mixed-type (continuous, discrete, and binary) variables. Furthermore, [9] also does not provide any differential privacy guarantees and instead uses ad-hoc notions of privacy which are only validated empirically.

Finally, it is worth remarking that it is known to be hard in the worst-case to generate private synthetic data [31] and so techniques such as GANs are necessary to address this challenge.

3 BACKGROUND

Let us denote the feature space by $\mathcal{X}$, the label space by $\mathcal{Y}$ and write $\mathcal{U} = \mathcal{X} \times \mathcal{Y}$. Let the dimension of $\mathcal{U}$ be d. Suppose that $\mathbf{X}$ and Y are random variables over $\mathcal{X}$ and $\mathcal{Y}$. We write $\mathbf{U} = (\mathbf{X}, Y)$ and $\mathbf{x}, y, \mathbf{u}$ to denote realizations of $\mathbf{X}, Y$ and $\mathbf{U}$, respectively. The dataset $\mathcal{D}$ consists of N samples of $\mathbf{u}$, assumed i.i.d. according to $\mathcal{P}_U$ denoted as $\mathcal{D} = \{\mathbf{u}_i\}_{i=1}^N = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$.

3.1 DIFFERENTIAL PRIVACY

We first provide some preliminaries on differential privacy [12] before describing PATE-GAN; we refer interested readers to [12] for a thorough exposition of differential privacy. We will denote an algorithm by $\mathcal{M}$, which takes as input a dataset $\mathcal{D}$ and outputs a value from some output space, $\mathcal{O}$.

Definition 1. *(Neighboring Datasets) Two datasets $\mathcal{D}, \mathcal{D}'$ are said to be neighboring if*

$$\exists x \in \mathcal{D} \text{ s.t. } \mathcal{D} \setminus \{x\} = \mathcal{D}'. \tag{1}$$

Definition 2. *(Differential Privacy) A* randomized *algorithm, $\mathcal{M}$, is (ϵ, δ)-differentially private if for all $\mathcal{S} \subset \mathcal{O}$ and for all neighboring datasets $\mathcal{D}, \mathcal{D}'$:*

$$\mathbb{P}(\mathcal{M}(\mathcal{D}) \in \mathcal{S}) \leq e^{\epsilon}\mathbb{P}(\mathcal{M}(\mathcal{D}') \in \mathcal{S}) + \delta \tag{2}$$

where $\mathbb{P}$ is taken with respect to the randomness of $\mathcal{M}$.

Differential privacy provides an intuitively understandable notion of privacy - a particular sample's inclusion or exclusion in the dataset does not change the probability of a particular outcome very much: it does so by a multiplicative factor of e^{ϵ} and an additive amount, δ.

The following theorem, a proof of which can be found in [12], allows us to move the burden of differential privacy to the discriminator; the differential privacy of the generator will follow by the theorem.

Theorem. *(Post-processing) Let $\mathcal{M}$ be an (ϵ, δ)-differentially private algorithm and let $f : \mathcal{O} \to \mathcal{O}'$ where $\mathcal{O}'$ is any arbitrary space. Then $f \circ \mathcal{M}$ is (ϵ, δ)-differentially private.*

3.2 PRIVATE AGGREGATION OF TEACHER ENSEMBLES (PATE)

In this section we describe the PATE mechanism first defined in [25] and later improved upon by [26]. The PATE mechanism provides a differentially private method for classification, a core component of the GAN framework; the discriminator is a classifier trained to identify whether samples are real/fake.

In order to build a differentially private classifier, the dataset is first divided into k *disjoint* subsets $\mathcal{D}_1, ..., \mathcal{D}_k$. k classifiers, $T_1, ..., T_k$ (referred to as *teachers*) are then trained *separately* on the k sub-datasets (i.e. T_i is only trained on $\mathcal{D}_i$). Given a new input feature vector $\mathbf{x}$ to classify, the

differentially private output is given by passing $\mathbf{x}$ to each of the k teachers, and then performing a *noisy* aggregation of the resulting outputs.

Formally, given the k teachers, m possible classes and an input feature vector, $\mathbf{x}$, set

$$n_j(\mathbf{x}) = |\{T_i : T_i(\mathbf{x}) = j\}| \text{ for } j = 1, ..., m \tag{3}$$

so that $n_j(\mathbf{x})$ is the number of teachers that output class j for $\mathbf{x}$. The output of the PATE$_\lambda$ mechanism for input $\mathbf{x}$ is then defined as

$$\text{PATE}_\lambda(\mathbf{x}) = \arg\max_{j \in [m]}(n_j(\mathbf{x}) + Y_j) \tag{4}$$

where $Y_1, ..., Y_m$ are i.i.d. $Lap(\lambda)$ random variables. The following result, found in [25], follows from [12].

Theorem. *The output of a single query to the PATE$_\lambda$ mechanism is $(\frac{1}{\lambda}, 0)$-differentially private.*

In order to apply this framework in the GAN framework, however, we require that the discriminator be differentiable, which the *output* of this classification mechanism is not (note that accessing the internal parameters of the teachers would violate differential privacy, the only thing we have access to in this case is the output). Instead, we draw on the PATE extension (also introduced in [25]) in which a *student* model is trained. This student model (after being trained) is free to access, not only its outputs given inputs but also its internal parameters. The model itself is differentially private.

Formally, the student, S, is a classifier that is trained by taking some *public, unlabelled* data, $\mathcal{P} = \{\mathbf{x}_i\}_{i=1}^K$, passing each sample, $\mathbf{x}_i$, through the (standard) PATE mechanism, to receive a differentially private label, $\hat{y}_i$, and forming a new (noisy-)teacher-labelled dataset $\hat{\mathcal{P}} = \{(\mathbf{x}_i, \hat{y}_i)\}_{i=1}^K$ on which the student is then trained.

Importantly, we can make the student differentiable - it can be modelled using any classifier, such as a neural net. Moreover, querying the student is "free" - there is no privacy cost associated with passing an input to the student and receiving an output, the only privacy cost is in acquiring the data on which to train the student. We state the following result which follows from the analysis in [25].

Theorem. *The student, S, trained on the dataset $\hat{\mathcal{P}}$ where labels were generated according to the PATE$_\lambda$ mechanism using $\lambda = \frac{K}{2\epsilon}$, is $(\epsilon, 0)$-differentially private* with respect to the original data $\mathcal{D}$.

4 Proposed Method: PATE-GAN

The proposed method builds on GAN and PATE frameworks. We replace the GAN discriminator with a PATE mechanism so that our discriminator is differentially private, but require the (differentiable) student version to allow back-propagation to the generator. We modify the implementation of the student, noting that the training paradigm presented in [25] is not appropriate for this setting due to the lack of publicly available data. Before training, we partition the dataset into k subsets, $\mathcal{D}_1, ..., \mathcal{D}_k$, with $|\mathcal{D}_i| = \frac{|\mathcal{D}|}{k}$ for $\forall i$.

4.1 Generator

The generator, G, is as in the standard GAN framework. Formally it is a function $G(\cdot; \theta_G) : [0,1]^d \to \mathcal{U}$, parametrized by θ_G that takes random noise, $\mathbf{z} \sim \text{Unif}([0,1]^d)$, as input and outputs a vector in $\mathcal{U} = \mathcal{X} \times \mathcal{Y}$. The generator will be trained to minimize its loss with respect to the *student*-discriminator. Given n i.i.d. samples of Unif$([0,1]^d)$, $\mathbf{z}_1, ..., \mathbf{z}_n$, the empirical loss of G at θ for fixed S is defined by

$$\mathcal{L}_G(\theta_G; S) = \sum_{j=1}^{n} \log(1 - S(G(\mathbf{z}_j; \theta_G))). \tag{5}$$

We will denote by $\mathcal{P}_G$ the distribution induced by G over $\mathcal{U}$.

4.2 Discriminator

In the standard GAN framework, there is a single discriminator, D, that is trained in a directly adversarial fashion with G, where at each iteration either G is trying to improve its loss with respect

to D or D is trying to improve its loss with respect to G. In our proposed model, however, we replace D with the PATE mechanism. This means we introduce k teacher-discriminators, $T^1, ..., T^k$, and a student discriminator, S. A noticeable difference is that the adversarial training is no longer symmetrical: the teachers are now being trained to improve their loss with respect to G but G is being trained to improve its loss with respect to the student S which in turn is being trained to improve its loss with respect to the teachers.

4.2.1 TEACHER-DISCRIMINATORS

Formally, the teacher-discriminators (which we will refer to simply as teachers) are functions $T_1(\cdot;\theta_T^1), ..., T_k(\cdot;\theta_T^k) : \mathcal{U} \to [0,1]$ each parametrized by θ_T^i. The teachers are given either a real sample from their corresponding partition of the dataset (i.e. T_i may receive a sample from $\mathcal{D}_i$) as input or a sample from the generator. The teachers are then trained to classify them.

Given n i.i.d. samples of Unif$([0,1]^d)$, $\mathbf{z}_1, ..., \mathbf{z}_n$, we define the empirical loss of teacher i with weights θ_T^i for fixed G by

$$\mathcal{L}_T^i(\theta_T^i) = -\Big[\sum_{\mathbf{u}\in\mathcal{D}_i} \log T_i(\mathbf{u};\theta_T^i) + \sum_{j=1}^{n} \log(1 - T_i(G(\mathbf{z}_j);\theta_T^i))\Big]. \tag{6}$$

Each teacher is trained in the same way the discriminator is trained in a standard GAN framework, except that here the teacher only ever sees its partition of the real data.

4.2.2 STUDENT-DISCRIMINATORS

The main innovation of our paper comes from our implementation of the student-discriminator (which we will refer to simply as the student) in this setting. The differential privacy guarantee provided by the standard student model is only with respect to the original data, $\mathcal{D}$, and not the public data, $\mathcal{P}$, used to train the student. In our setting, where the entire focus is on generating synthetic data *because no data is publicly available*, we must propose a novel methodology to train the student without public data.

We first note, that the student training paradigm described in [25] would involve training the student using data similar to that used to train the generator - i.e. by taking an equal number of samples from each and then labelling those using the standard PATE$_\lambda$ mechanism (where here "labelling" refers to assigning them a real/fake label - not the label y present in the data). We consider the implications of training the student on teacher-labelled generated samples only.

We first observe that during training of the generator, the discriminator is only evaluated on samples from the generator itself, and not the real data, so by training the student only on generated samples we are in fact training it on the distribution we need it to perform well on. However, we note that if the student only sees unrealistic samples from the generator (i.e. generated samples that most teachers label as fake), then the student will not contain any information that the generator can use to improve its generated samples. It is therefore important that some of the generated samples the student is trained on are realistic. We then note that if $\text{Supp}(\mathcal{P}_U) \subset \text{Supp}(\mathcal{P}_G)$ then some of the generated samples will be realistic.

In order to ensure $\text{Supp}(\mathcal{P}_U) \subset \text{Supp}(\mathcal{P}_G)$, we normalize the data into $[0,1]^d$ and then initialize the parameters of the generator randomly using Xavier initialization. It follows that $\text{Supp}(\mathcal{P}) \subset [0,1]^d \subset G([0,1]^d) = G(\text{Supp}(\mathbf{Z})) = \text{Supp}(G(\mathbf{Z}))$ when $\mathbf{Z} \sim \text{Unif}([0,1]^d)$.

We create our training data for the student by taking n i.i.d. samples of Unif$([0,1]^d)$, $\mathbf{z}_1, ..., \mathbf{z}_n$, generating n samples using the generator, $\hat{\mathbf{u}}_1, ..., \hat{\mathbf{u}}_n$ with $\hat{\mathbf{u}}_j = G(\mathbf{z}_j)$, and using the teachers to label these using PATE$_\lambda$, setting $r_j = \text{PATE}_\lambda(\hat{\mathbf{u}}_j)$. We train the student, $S(\cdot;\theta_S) : \mathcal{U} \to [0,1]$, to maximize the standard cross-entropy loss on this teacher-labelled data, i.e.

$$\mathcal{L}_S(\theta_S) = \sum_{j=1}^{n} r_j \log S(\hat{\mathbf{u}}_j;\theta_S) + (1 - r_j)\log(1 - S(\hat{\mathbf{u}}_j;\theta_S)). \tag{7}$$

Although a priori the above mechanism does not appear to depend on the number of teachers, it should be noted that for fixed λ, more teachers results in the teacher-labelled dataset being less

noisy - the noise being added is smaller relative to the counts n_j. This introduces a trade-off - for a small number of teachers, the noise may be too large and thus render the output meaningless; with a larger number of teachers, less data can be used to train each teacher, which may also render the output meaningless, even though the noise has a smaller effect. Finding the right balance in this problem is key. In our experiments, we use d real and d generated samples to train each teacher where d is the dimension of the input space. Although the utility of a single teacher may be low, by aggregating (even noisily) the resulting classifier actually has high utility. Moreover, by using a minimal number of samples for each teacher, the effect of any individual sample on the output is small (because there are more teachers and each sample can effect at most 1 teacher) which means that our differential privacy guarantees are tighter - if we used fewer teachers, the mechanism still assumes that, in the worst case, the presence (or absence) of a single sample can completely flip a teacher's vote and so we still need to add the same noise.

We train G, $T^1, ..., T^k$ and S iteratively[1], with each iteration of G consisting of first performing n_T updates on all teachers, then performing n_S updates of the student. We perform generator iterations until our privacy constraint, ϵ, has been reached. A block diagram of PATE-GAN can be found in the Appendix.

To calculate the privacy of our algorithm we use the moments accountant method given in [25] to derive a data-dependent privacy guarantee at run-time. Details of its definition, and key results we use can be found in the Appendix. We denote the moments accountant of PATE-GAN by $\alpha(l)$. The moments accountant allows us to more tightly bound the total privacy cost of our mechanism than standard composition theorems would, and moreover attributes a lower privacy cost to accessing the noisy aggregation of the teachers when the teachers have a stronger consensus with the intuition being that when the teachers have a strong consensus, a single teacher (and therefore a single sample) has a much lower influence on the output than when the votes (n_0 and n_1) are close. Pseudo-code for PATE-GAN can be found in Algorithm 1.

We now state the main theorem of the paper, which follows from the theory in [25].

Theorem 1. *Algorithm 1, which takes as input $\delta > 0$, a dataset, $\mathcal{D}$, and outputs G and ϵ is (ϵ, δ)-differentially private.*

The proof relies on applying the post-processing theorem where the discriminator corresponds to the mechanism $\mathcal{M}$ which takes outputs in $\mathcal{O}$ (in our case this corresponds to the weights of the discriminator), and the generator corresponds to the function f which maps from $\mathcal{O}$ to $\mathcal{O}$ (which corresponds to the weights of the generator). For full details of the proof and further details of the theory required for it, see the Appendix.

5 EXPERIMENTS

In this section, we use a real-world Kaggle dataset (Credit card fraud detection dataset [11]) to evaluate PATE-GAN against the state-of-the-art benchmark (DPGAN [32]). In addition, we provide high-level (average) results for five additional datasets (with various characteristics): MAGGIC [27], UNOS-Heart wait-list [7], Kaggle cervical cancer dataset [16], UCI ISOLET dataset and UCI Epileptic Seizure Recognition dataset. A more detailed breakdown of the results for these datasets is given in the Appendix. Details of all six datasets can be also found in the Appendix.

5.1 EXPERIMENTAL SETTINGS

To empirically validate the quality of the generated dataset we introduce three different training-testing settings. *Setting A:* train the predictive models on the real training set, test the performance of the models on the real testing set. *Setting B:* train on the synthetic training set, test on the real testing set ([15]), *Setting C:* train on the synthetic training set, test on the synthetic testing set. Note that the training set and the testing set are disjoint in both the real and synthetic datasets.

We are interested in two comparisons. If we see a high predictive performance on the real data for models that were trained on synthetic data (Setting B), we can infer that the synthetic data has

[1]The teachers can be trained in parallel.

Algorithm 1 Pseudo-code of PATE-GAN

Input: δ, $\mathcal{D}$, n_T, n_S, batch size n, number of teachers k, noise size λ
Initialize: θ_G, $\theta_T^1, ..., \theta_T^k$, θ_S, $\alpha(l) = 0$ for $l = 1, ..., L$
Partition dataset into k subsets $\mathcal{D}_1, ..., \mathcal{D}_k$ of size $\frac{|\mathcal{D}|}{k}$
while $\hat{\epsilon} < \epsilon$ **do**
  for $t_2 = 1, ..., n_T$ **do**
    Sample $\mathbf{z}_1, ..., \mathbf{z}_n \overset{\text{i.i.d.}}{\sim} \mathcal{P}_\mathcal{Z}$
    for $i = 1, ..., k$ **do**
      Sample $\mathbf{u}_1, ..., \mathbf{u}_n \overset{\text{i.i.d.}}{\sim} \mathcal{D}_i$
      Update teacher, T_i, using SGD
      $\nabla_{\theta_T^i} - \left[\sum_{j=1}^{d} \log(T_i(\mathbf{u}_j)) + \log(1 - T_i(G(\mathbf{z}_j)))\right]$
  for $t_3 = 1, ..., n_S$ **do**
    Sample $\mathbf{z}_1, ..., \mathbf{z}_n \overset{\text{i.i.d.}}{\sim} \mathcal{P}_\mathcal{Z}$
    for $j = 1, ..., n$ **do**
      $\hat{\mathbf{u}}_j \leftarrow G(\mathbf{z}_j)$
      $r_j \leftarrow \text{PATE}_\lambda(\hat{\mathbf{u}}_i)$ for $j = 1, ..., n$
      Update moments accountant
      $q \leftarrow \frac{2+\lambda|n_0 - n_1|}{4\exp(\lambda|n_0 - n_1|)}$
      for $l = 1, ..., L$ **do**
        $\alpha(l) \leftarrow \alpha(l) + \min\{2\lambda^2 l(l+1), \log((1-q)\left(\frac{1-q}{1-e^{2\lambda}q}\right)^l + qe^{2\lambda l})\}$
    Update the student, S, using SGD
    $\nabla_{\theta_S} - \sum_{j=1}^{n} r_j \log S(\hat{\mathbf{u}}_j) + (1 - r_j)\log(1 - S(\hat{\mathbf{u}}_j))$
  Sample $\mathbf{z}_1, ..., \mathbf{z}_n \overset{\text{i.i.d.}}{\sim} \mathcal{P}_\mathcal{Z}$
  Update the generator, G, using SGD
  $\nabla_{\theta_G}\left[\sum_{i=1}^{n} \log(1 - S(G(\mathbf{z}_i))\right]$
  $\hat{\epsilon} \leftarrow \min_l \frac{\alpha(l) + \log(\frac{1}{\delta})}{l}$
Output: G

captured the relationship between features and labels well. Moreover, synthetic data that does well in this setting can be used to train models without ever seeing the real data.

On the other hand, when we consider synthetic data for use in competitions such as Kaggle, we need synthetic data that allows researchers to do meaningful comparisons on the synthetic data. In this setting, the researchers will only be able to use the synthetic data as both the training and testing set, and will need to develop their algorithms using results on the synthetic data. Now it becomes important that the relative performance of two algorithms when trained and tested on the synthetic data (Setting C), is similar to their relative performance when trained and tested on the real data (Setting A). A simple requirement would be that if model 1 is better than model 2 on the real data, then model 1 is better than model 2 on the synthetic data. This allows researchers to use the synthetic data to choose the best method(s) to try on the real data (or rather to give to the data-holder to try on the real data).

For both comparisons, we use 12 different predictive models, shown in Table 1. We use two performance metrics to measure the capability of each model in predicting the label: (1) area under the receiver operating characteristics curve (AUROC), (2) area under the precision recall curve (AUPRC). Throughout the experiments we fix $\delta = 10^{-5}$ for use as input to PATE-GAN and DPGAN. We also report the performance of the original GAN framework ("GAN"), which serves to indicate an upper bound on performance and allows us to see how much performance is lost due to the two differential privacy mechanisms (PATE-GAN and DPGAN). The details of hyper-parameter optimization and benchmark implementations can be found in the Appendix.

	AUROC			AUPRC		
	GAN	**PATE-GAN**	DPGAN	GAN	**PATE-GAN**	DPGAN
Logistic Regression	0.8950	0.8728	0.8720	0.4069	0.3907	0.3923
Random Forests [5]	0.9075	0.8980	0.8730	0.3219	0.3157	0.2926
Gaussian Naive Bayes [29]	0.8861	0.8817	0.8522	0.1963	0.1858	0.1601
Bernoulli Naive Bayes [29]	0.8997	0.8968	0.8891	0.2169	0.2099	0.2069
Linear SVM [10]	0.7611	0.7523	0.7502	0.4473	0.4466	0.4464
Decision Tree [28]	0.9102	0.9011	0.8647	0.4071	0.3978	0.3672
LDA [3]	0.8710	0.8510	0.8487	0.1956	0.1852	0.1788
AdaBoost [17]	0.9143	0.8952	0.8809	0.4530	0.4366	0.4234
Bagging [4]	0.8951	0.8877	0.8657	0.3303	0.3221	0.3073
GBM [18]	0.8848	0.8709	0.8499	0.3057	0.2974	0.2773
Multi-layer Perceptron	0.9086	0.8925	0.8787	0.4790	0.4693	0.4600
XgBoost [8]	0.9058	0.8904	0.8637	0.3837	0.3700	0.3440
Average	**0.8866**	**0.8737**	**0.8578**	**0.3453**	**0.3351**	**0.3219**

Table 1: Performance comparison of 12 different predictive models in Setting B (trained on synthetic, tested on real) in terms of AUROC and AUPRC (the generators of PATE-GAN and DPGAN are $(1, 10^{-5})$-differentially private).

5.2 Results with Setting B

In this subsection, we evaluate PATE-GAN and DPGAN in Setting B (trained on synthetic, tested on real) to understand whether or not the models are capturing the feature-label relationships well. Intuitively, if a synthetic dataset is such that a model trained on it performs well when performance is measured on real data, then the relationship between feature and label in the synthetic data is similar to that in the real data. In Table 1 we give the results for the Kaggle Credit dataset for all 12 predictive models. In Table 2, we give the performance on each dataset averaged across the 12 methods for each of the 6 datasets. A breakdown of the performance of each predictive model for each dataset can be found in the Appendix. Across all datasets, we see that PATE-GAN is capable of generating synthetic samples that better preserve the feature-label relationship (according to AUROC and AUPRC) than DPGAN.

Datasets	AUROC			AUPRC		
	GAN	PATE-GAN	DPGAN	GAN	PATE-GAN	DPGAN
Kaggle Credit	0.8866	0.8737	0.8578	0.3453	0.3351	0.3219
MAGGIC	0.6574	0.6446	0.6286	0.3054	0.2952	0.2820
UNOS	0.6277	0.5996	0.5552	0.6554	0.6282	0.5862
Kaggle Cervical Cancer	0.9268	0.9108	0.8699	0.5994	0.5460	0.4851
UCI ISOLET	0.8171	0.6399	0.5577	0.5561	0.2953	0.2146
UCI Epileptic Seizure Recognition	0.9173	0.7681	0.6718	0.8133	0.6512	0.5369

Table 2: Performance comparison of 12 different predictive models in Setting B (trained on synthetic, tested on real) in terms of AUROC and AUPRC (the generators of PATE-GAN and DPGAN are $(1, 10^{-5})$-differentially private) over 6 different datasets. GAN is (∞, ∞)-differentially private and is given to indicate an upper bound of PATE-GAN and DPGAN.

We note that the performance of all models, including the original GAN model (i.e. PATE-GAN - or equivalently DPGAN - with (∞, ∞)-differential privacy) in the high dimensional UCI ISOLET and UCI Epileptic Seizure Recognition datasets is lower than in lower dimensional datasets (when

compared to the baseline AUROC and AUPRC found in the Appendix). We do, however, see that both PATE-GAN and DPGAN show more significant decreases in performance than the original GAN in these high-dimensional settings. In the case of PATE-GAN, we believe this may be due to the fact that the student discriminator is trained only using data from the generator, and therefore requires that some of the generated data look somewhat realistic from the start, which is a harder requirement to satisfy as the data has more dimensions. On the other hand, in DPGAN, noise must be added to each component of the gradient (of the discriminator) and so in higher dimensions the norm of the noise added is larger. Note that in PATE-GAN, noise is added only to the teacher outputs, whose dimension (typically 1) does not depend on the dimension of the input data, and so this phenomena does not present itself in PATE-GAN. The results on both the UCI datasets would suggest that the loss from increasing noise norm (for DPGAN) is greater than from difficulty in randomly generating realistic samples (for PATE-GAN).

5.3 VARYING THE PRIVACY CONSTRAINT (ϵ)

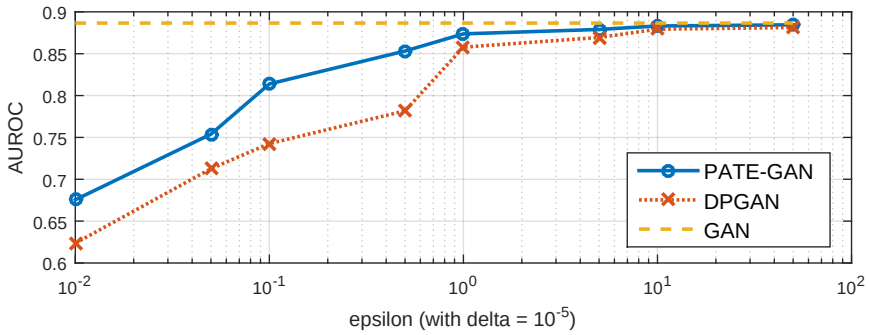


Figure 1: Average AUROC performance across 12 different predictive models trained on the synthetic dataset generated by PATE-GAN and DPGAN with various ϵ (with $\delta = 10^{-5}$) (Setting B).

In Fig. 1, we investigate the trade off between privacy constraint and utility. In the table we report the average performance of AUROC over the 12 different predictive models for PATE-GAN and the benchmark for various ϵ (with $\delta = 10^{-5}$). As can be seen in Fig. 1, PATE-GAN is consistently better than DPGAN over the entire range of tested ϵ. We believe this is because the PATE mechanism allows us to more tightly bound the influence of a single sample on the discriminator, and hence we can provide tighter differential privacy guarantees - when the differential privacy guarantee is fixed, this results in higher quality synthetic data. Of course, as we increase ϵ (i.e. decrease the required privacy) both methods converge to the performance of GAN and the increase in performance of PATE-GAN over DPGAN becomes smaller.

5.4 SETTING A VS SETTING C: PRESERVING THE RANKING OF PREDICTIVE MODELS

As discussed at the beginning of this section, it is important that a synthetic dataset respects the ranking of models (in terms of their prediction performances) [21]. To evaluate this, we now introduce a new metric, which we refer to as the Synthetic Ranking Agreement (SRA). Suppose that we have L predictive models, $f_1, f_2, ..., f_L$[2]. Furthermore, suppose that the performance of model i when trained and tested on the real data (Setting A) is $A_i \in \mathbb{R}$ and that the performance of model i when trained and tested on the synthetic data (Setting C) is $C_i \in \mathbb{R}$. Then we define the Synthetic Ranking Agreement by

$$\textbf{SRA}(\{A_i\}_{i=1}^L, \{C_i\}_{i=1}^L) = \frac{1}{L(L-1)} \sum_{j=1}^{L} \sum_{k \neq j} \mathbb{I}\Big((A_j - A_k) \times (C_j - C_k) > 0\Big) \tag{8}$$

where $\mathbb{I}$ is an indicator function. Note that the summand is 1 when the ordering of algorithms j and k are the same in both settings, and is 0 when the ordering in one setting differs from the ordering in the other.

[2]For the results in Table 3, we use the same 12 predictive models as used in Table 1

	PATE-GAN	**DPGAN**		**PATE-GAN**	**DPGAN**
$\epsilon = 0.01$	0.6909	0.5273	$\epsilon = 1$	0.8364	0.8000
$\epsilon = 0.05$	0.7455	0.6909	$\epsilon = 5$	0.8909	0.8364
$\epsilon = 0.1$	0.7818	0.7455	$\epsilon = 10$	0.9091	0.8909
$\epsilon = 0.5$	0.8000	0.7818	$\epsilon = 50$	0.9091	0.9091

Table 3: Synthetic Ranking Probability of PATE-GAN and the benchmark when comparing Setting A and Setting C for various ϵ (with $\delta = 10^{-5}$) in terms of AUROC. The Synthetic Ranking Agreement of Original GAN is 0.9091, which is also attained by both PATE-GAN and DPGAN for $\epsilon = 50$.

We compare the SRA of PATE-GAN and the benchmark for various ϵ (with $\delta = 10^{-5}$)[3]. As can be seen in Table 3, PATE-GAN achieves the best SRA across all values of ϵ.

In the Appendix, we perform a similar experiment in which we compare the ranking of features by their importance (determined by their absolute Pearson correlation coefficient with the label) on the original dataset and on the synthetic dataset (generated by PATE-GAN and the benchmark) and report the results using a metric that is identical to SRA, with the model performances ($\{A_i\}, \{C_i\}$) substituted for feature importances.

5.5 Quantitative analysis on the number of teachers

The number of teachers is a hyper-parameter of PATE-GAN and we choose the number of teachers among $\{N/10, N/50, N/100, N/500, N/1000, N/5000, N/10000\}$ where N is the total number of samples. As we described in the previous section, there is a trade-off between number of teachers and the corresponding quality of the synthetic data. Table 4 quantitatively shows the trade-off between the number of teachers and the performance (in terms of both AUROC and AUPRC).

# of teachers	$N/10$	$N/50$	$N/100$	$N/500$	**N/1000**	$N/5000$	$N/10000$
AUROC	0.5425	0.6398	0.7638	0.8343	**0.8737**	0.8655	0.8282
AUPRC	0.1273	0.2484	0.2900	0.3184	**0.3351**	0.3278	0.3092

Table 4: Trade-off between the number of teachers and the performances (AUROC, AUPRC)

6 Discussion

In this paper we introduced a novel methodology for generating differentially private synthetic data. Through several experiments we demonstrated the ability of our method to produce high quality synthetic data while being able to give strict differential privacy guarantees.

In order to apply PATE to the GAN setting, we needed to use the original GAN framework. Extending PATE to the regression setting so that, for example, a Wasserstein GAN can be used instead, is an open and interesting question, and a potential direction for future research.

Acknowledgement

The authors would like to thank the reviewers for their helpful comments. The research presented in this paper was supported by the Office of Naval Research (ONR) and the NSF (Grant number: ECCS1462245, ECCS1533983, and ECCS1407712).

[3]The ordering of models according to Table 1 is in fact quite consistent - the average agreed ranking probability (now applied to different folds of the data, rather than real vs. synthetic data) is 0.9273 (for AUROC). The rankings used are therefore sufficiently stable for this to be a meaningful metric.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318. ACM, 2016.

[2] Brett K Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, and Casey S Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *BioRxiv*, pp. 159756, 2017.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[4] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] Anna L Buczak, Steven Babin, and Linda Moniz. Data-driven approach for creating synthetic electronic medical records. *BMC medical informatics and decision making*, 10(1):59, 2010.

[7] J Michael Cecka and Paul I Terasaki. The unos scientific renal transplant registry. *Clinical transplants*, pp. 1–18, 1993.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794. ACM, 2016.

[9] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete electronic health records using generative adversarial networks. *arXiv preprint arXiv:1703.06490*, 2017.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

[11] Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *Computational Intelligence, 2015 IEEE Symposium Series on*, pp. 159–166. IEEE, 2015.

[12] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[13] Khaled El Emam, David Buckeridge, Robyn Tamblyn, Angelica Neisa, Elizabeth Jonker, and Aman Verma. The re-identification risk of canadians from longitudinal demographics. *BMC medical informatics and decision making*, 11(1):46, 2011.

[14] Yaniv Erlich and Arvind Narayanan. Routes for breaching and protecting genetic privacy. *Nature Reviews Genetics*, 15(6):409–421, 2014.

[15] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[16] Kelwin Fernandes, Jaime S Cardoso, and Jessica Fernandes. Transfer learning with partial observability applied to cervical cancer screening. In *Iberian conference on pattern recognition and image analysis*, pp. 243–250. Springer, 2017.

[17] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *Icml*, volume 96, pp. 148–156. Bari, Italy, 1996.

[18] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[20] J Henry, Y Pylypchuk, T Searcy, and V Patel. Adoption of electronic health record systems among us non-federal acute care hospitals: 2008–2015. may 2016. *Accessed via:¡ https://dashboard. healthit. gov/evaluations/data-briefs/non-federal-acute-care-hospital-ehr-adoption-2008-2015. php¿. Accessed on June*, 21, 2017.

[21] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Measuring the quality of synthetic data for use in competitions. *arXiv preprint arXiv:1806.11345*, 2018.

[22] Bradley Malin and Latanya Sweeney. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. *Journal of biomedical informatics*, 37(3):179–192, 2004.

[23] Scott McLachlan, Kudakwashe Dube, and Thomas Gallagher. Using the caremap with health incidents statistics for generating the realistic synthetic electronic healthcare record. In *Healthcare Informatics (ICHI), 2016 IEEE International Conference on*, pp. 439–448. IEEE, 2016.

[24] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pp. 111–125. IEEE, 2008.

[25] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[26] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[27] Stuart J Pocock, Cono A Ariti, John JV McMurray, Aldo Maggioni, Lars Køber, Iain B Squire, Karl Swedberg, Joanna Dobson, Katrina K Poppe, Gillian A Whalley, et al. Predicting survival in heart failure: a risk score based on 39 372 patients from 30 studies. *European heart journal*, 34(19):1404–1413, 2012.

[28] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[29] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pp. 41–46. IBM, 2001.

[30] Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. *The Journal of Law, Medicine & Ethics*, 25(2-3):98–110, 1997.

[31] Jonathan Ullman and Salil Vadhan. Pcps and the hardness of generating private synthetic data. In *Theory of Cryptography Conference*, pp. 400–416. Springer, 2011.

[32] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

APPENDIX

THEORY REQUIRED FOR THEOREM 1

Theorem 2. *Algorithm 1, which takes as input $\delta > 0$, a dataset, $\mathcal{D}$, and outputs G and ϵ is (ϵ, δ)-differentially private.*

In order to prove our theorem, we define the moments accountant [1] and state the theorems that the data-dependent privacy guarantees of the PATE mechanism rely on. For proofs of the results below, see [25] and the references therein.

Definition 3. *(Privacy Loss) Let $\mathcal{M}$ be a randomized algorithm taking outputs in a space $\mathcal{O}$ and $\mathcal{D}, \mathcal{D}'$ be neighbouring datasets. Let aux denote an auxiliary input. For an outcome $o \in \mathcal{O}$, the privacy loss at o is defined as:*

$$c(o; \mathcal{M}, aux, \mathcal{D}, \mathcal{D}') = \log \frac{\mathbb{P}(\mathcal{M}(aux, \mathcal{D}) = o)}{\mathbb{P}(\mathcal{M}(aux, \mathcal{D}') = o)}. \tag{9}$$

The privacy loss random variable $C(\mathcal{M}, aux, \mathcal{D}, \mathcal{D}')$ is defined as $c(\mathcal{M}(\mathcal{D}); \mathcal{M}, aux, \mathcal{D}, \mathcal{D}')$, i.e. the random variable defined by evaluating the privacy loss at an outcome sampled from $\mathcal{M}(\mathcal{D})$.

Definition 4. *(Moments accountant) Let $\mathcal{M}$ be a randomized algorithm. The moments accountant is defined as:*

$$\alpha_{\mathcal{M}}(l) = \max_{aux, \mathcal{D}, \mathcal{D}'} \alpha_{\mathcal{M}}(l; aux, \mathcal{D}, \mathcal{D}') \tag{10}$$

where $\alpha_{\mathcal{M}}(l; aux, \mathcal{D}, \mathcal{D}') = \log \mathbb{E}(\exp(lC(\mathcal{M}, aux, \mathcal{D}, \mathcal{D}')))$ is the moment generating function of the privacy loss random variable and the $\max$ is taken over neighbouring datasets $\mathcal{D}, \mathcal{D}'$.

Theorem 3. *(Composability) Suppose that an algorithm $\mathcal{M}$ consists of a sequence of adaptive algorithms $\mathcal{M}_1, ..., \mathcal{M}_k$ where $\mathcal{M}_i$ outputs in $\mathcal{O}_i$ and takes inputs from $\Pi_{j=1}^{i-1} \mathcal{O}_j$ as well as the dataset $\mathcal{D}$. Then for any output sequence $o_1, ..., o_{k-1}$ and any l*

$$\alpha_{\mathcal{M}}(l; \mathcal{D}, \mathcal{D}') = \sum_{i=1}^{k} \alpha_{\mathcal{M}_i}(l; o_1, ..., o_{i-1}, \mathcal{D}, \mathcal{D}') \tag{11}$$

where $\alpha_{\mathcal{M}}$ is conditioned on each $\mathcal{M}_i$'s output being o_i.

Theorem 4. *(Tail bound) Let $\mathcal{M}$ be a randomized algorithm. For any $\epsilon > 0$, $\mathcal{M}$ is (ϵ, δ)-differentially private for*

$$\delta = \min_{l} \exp(\alpha_{\mathcal{M}}(l) - l\epsilon). \tag{12}$$

The following theorem combines Theorems 2, 3 and Lemma 4 from [25].

Theorem 5. *(Data-dependent privacy guarantee for PATE) Let $\mathcal{M}$ be the PATE mechanism defined in Section 3 of the paper. Let n_0, n_1 be as defined in Equation 3 of the paper. Let $q = \frac{2+\lambda|n_0-n_1|}{4\exp(\lambda|n_0-n_1|)}$. Then*

$$\alpha_{\mathcal{M}}(l) \leq \min\{2\lambda^2 l(l+1), \log((1-q)\left(\frac{1-q}{1-e^{2\lambda}q}\right)^l + qe^{2\lambda l})\}. \tag{13}$$

Proof of Theorem 2. We use Theorem 5 to bound the moments accountant for each query to the PATE mechanism during the training of our algorithm (i.e. each time a generated sample is labeled by the teachers). Theorem 3 then allows us to sum the individual bounds for each query to bound the moments accountant of the entire algorithm. Theorem 4 then allows us to derive a value for ϵ given δ. □

DATA DESCRIPTIONS

KAGGLE CREDIT CARD FRAUD DETECTION DATA DESCRIPTION

The Kaggle credit card fraud detection dataset [11] contains transactions made by credit cards in September 2013 by European cardholders and the label is whether or not the transaction is fraudulent. The total number of features is 29 (binary: 0, continuous: 29) and the number of samples in this dataset is 284,807. Among the 284,807 samples, 492 (0.2%) samples are fraudulent transactions.

MAGGIC DATA DESCRIPTION

The Meta-analysis Global Group in Chronic Heart Failure (MAGGIC) dataset [27] is a collection of 30 different datasets from 30 different medical studies containing patients who experienced heart failure. We set the label of each patient as 1-year all-cause mortality, excluding all patients who are censored before 1-year. The total number of features is 29 (binary: 20, continuous: 9) and the number of patients in this dataset is 30,389. Among the 30,389 patients, 5,723 (18.8%) patients died within 1 year.

UNOS DATA DESCRIPTION

The United Network for Organ Transplantation (UNOS) dataset [7] provides information about all patients in the U.S. who have received a transplantation or were on the wait-list during the period 1985-2015. In this paper, we focus on the patients who were on the heart transplant wait-list. The objective is to predict 1-year all-cause mortality. The total number of features is 20 (binary: 18, continuous: 2) and the number of patients in this dataset is 23,706. Among the 23,706 patients, 12,606 (53.2%) patients died within 1 year.

KAGGLE CERVICAL CANCER DATA DESCRIPTION

The Kaggle cervical cancer dataset [16] was collected at 'Hospital Universitario de Caracas' in Caracas, Venezuela. It contains demographic information, habits, and historic medical records. The total number of features is 35 (binary: 24, continuous: 11) and the number of patients in this dataset is 858. Among the 858 patients, 55 (6.4%) patients have positive biopsy.

UCI ISOLET DATA DESCRIPTION

The UCI ISOLET dataset `https://archive.ics.uci.edu/ml/datasets/isolet` was generated by speaking the name of each letter of the alphabet. The task is to classify each spoken letter as either a vowel or a consonant (binary classification). The total number of features is 617 and the number of samples in this dataset is 7797. Among the 7797 samples, 1500 (19.2%) samples are vowels.

UCI EPILEPTIC SEIZURE RECOGNITION DATA DESCRIPTION

The UCI Epileptic Seizure Recognition dataset `https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition` was generated by recording brain activity. The task is to classify activity as seizure activity (binary classification). The total number of features is 179 and the number of samples in this dataset is 11500. Among the 11500 samples, 2300 (20.0%) samples correspond to seizure activity.

DATA SUMMARY AND SETTING A PERFORMANCE

Table 5 summarises the 6 datasets we use and provides a baseline performance for a predictive model on each dataset - recall that Setting A refers to training and testing on the real data. The AUROC and AUPRC in this setting are upper bounds on the AUROC and AUPRC we could hope to achieve in Setting B.

Datasets	No of samples	No of features	AUROC	AUPRC
Kaggle Credit	284807	29	0.9438	0.7020
MAGGIC	30389	29	0.7069	0.3638
UNOS	23706	20	0.6416	0.6677
Kaggle Cervical cancer	858	35	0.9354	0.6314
UCI ISOLET	7797	617	0.9671	0.8758
UCI Epileptic Seizure Recognition	11500	179	0.9809	0.9511

Table 5: No of samples, No of features, Average AUROC and AUPRC performance across 12 different predictive models trained and tested on the real data (Setting A) for the 6 datasets: Kaggle Credit, MAGGIC, UNOS, Kaggle Cervical Cancer, UCI ISOLET, UCI Epileptic Seizure Recognition.

HYPER-PARAMETER OPTIMIZATION

In all experiments, the depth of the generator and discriminator (student-discriminator in our case) in both PATE-GAN and the DPGAN benchmark [32] is set to 3. The depth of the teacher discriminators is set to 1. The number of hidden nodes in each layer is d, $d/2$ and d (where d is the feature dimension), respectively. We use *relu* as the activation functions of each layer except for the output layer where we use the *sigmoid* activation function and the batch size is 64 for both the generator and discriminator. We set $n_T = n_S = 5$. Using cross validation, we select the number of teachers, k, among N/10 N/50 N/100 N/500 N/1000 N/5000 N/10000. The learning rate is 10^{-4} and we use Adam Optimizer to minimize the loss function.

We use tensorflow to implement PATE-GAN and DPGAN. For DPGAN we use the code from the following link: `https://github.com/illidanlab`. We use the sklearn package in python to implement the 12 predictive models: Logistic Regression (*LogisticRegression*), Random Forests (*RandomForestClassifier*), Gaussian Naive Bayes (*GaussianNB*), Bernoulli Naive Bayes (*BernoulliNB*), Linear Support Vector Machine (*svm*), Decision Tree (*DecisionTree*), Linear Discriminant Analysis Classifier (*LinearDiscriminantAnalysis*), Adaptive Boosting (AdaBoost) (*AdaBoostClassifier*), Bootstrap Aggregating (Bagging) (*BaggingClassifier*), Gradient Boosting Machine (GBM) (*GradientBoostingClassifier*), Multi-layer Perceptron (*MLPClassifier*), and XgBoost (*XGBoostRegressor*).

ADDITIONAL RESULTS

VARYING THE PRIVACY CONSTRAINT (ϵ) IN TERMS OF AUPRC

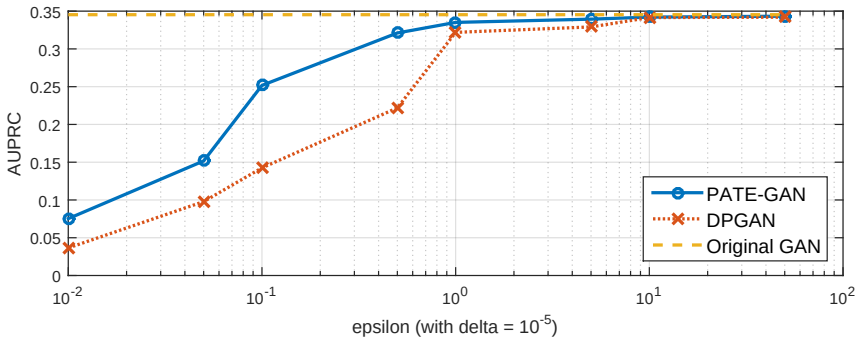


Figure 2: Average AUPRC performance across 12 different predictive models trained on the synthetic dataset generated by PATE-GAN and DPGAN with various ϵ (with $\delta = 10^{-5}$) (Setting B).

Similar to Fig. 1 in the main manuscript, Fig. 2 shows the trade off between the privacy constraint and utility, where utility is now measured in terms of AUPRC (rather than AUROC). We report the average performance in terms of AUPRC over the 12 different predictive models for PATE-GAN and the benchmark for various ϵ (with $\delta = 10^{-5}$). As can be seen in Fig. 2, PATE-GAN consistently outperforms DPGAN over the entire range of tested ϵ in terms of AUPRC as well.

PRESERVING THE RANKING OF VARIABLE IMPORTANCE IN KAGGLE CREDIT DATASET

We compare the ranking of variables by their importance (according to absolute Pearson correlation coefficient with the label) on the original dataset and on both synthetic datasets. We report the results using agreed ranking probability. As can be seen in Table 6, PATE-GAN achieves consistently better agreed ranking probability across all values of tested ϵ (with $\delta = 10^{-5}$).

	PATE-GAN	**DPGAN**		**PATE-GAN**	**DPGAN**
$\epsilon = 0.01$	0.8810	0.7963	$\epsilon = 1$	0.9048	0.8783
$\epsilon = 0.05$	0.8968	0.8148	$\epsilon = 5$	0.9127	0.8915
$\epsilon = 0.1$	0.8968	0.8333	$\epsilon = 10$	0.9153	0.8942
$\epsilon = 0.5$	0.9021	0.8545	$\epsilon = 50$	0.9153	0.9021

Table 6: Agreed ranking probability of PATE-GAN and the benchmark to order the features by variable importance in terms of absolute Pearson correlation coefficient

HIGH-DIMENSIONAL RESULTS: UCI ISOLET AND UCI EPILEPTIC SEIZURE RECOGNITION

(ϵ, δ)	**AUROC**			**AUPRC**		
	GAN	PATE-GAN	DPGAN	GAN	PATE-GAN	DPGAN
$(10, 10^{-5})$	0.8171	0.7688	0.7390	0.5561	0.4734	0.3831
$(1, 10^{-5})$		0.6399	0.5577		0.2953	0.2146

Table 7: Average AUROC and AUPRC performance of 12 different predictive models trained on the synthetic datasets for $\epsilon = 1, 10$ with $\delta = 10^{-5}$ - Setting B using UCI ISOLET dataset. GAN is (∞, ∞)-differentially private and is given to indicate an upper bound of PATE-GAN and DPGAN.

(ϵ, δ)	AUROC			AUPRC		
	GAN	PATE-GAN	DPGAN	GAN	PATE-GAN	DPGAN
$(10, 10^{-5})$	0.9173	0.8718	0.8189	0.8133	0.7662	0.7201
$(1, 10^{-5})$		0.7681	0.6718		0.6512	0.5369

Table 8: Average AUROC and AUPRC performance across 12 different predictive models trained on the synthetic datasets with various $\epsilon = 1, 10$ with $\delta = 10^{-5}$ - Setting B using UCI Epileptic Seizure Recognition dataset. GAN is (∞, ∞)-differentially private and is given to indicate an upper bound of PATE-GAN and DPGAN.

MAGGIC DATASET RESULT

	AUROC			AUPRC		
	GAN	**PATE-GAN**	**DPGAN**	GAN	**PATE-GAN**	**DPGAN**
Logistic Regression	0.6645	0.6413	0.6415	0.3113	0.2951	0.2967
Random Forests	0.6492	0.6397	0.6147	0.2953	0.2891	0.2660
Gaussian Naive Bayes	0.6770	0.6726	0.6431	0.3258	0.3153	0.2896
Bernoulli Naive Bayes	0.6647	0.6618	0.6541	0.3008	0.2938	0.2908
Linear SVM	0.6410	0.6301	0.6322	0.2911	0.2904	0.2902
Decision Tree	0.6689	0.6598	0.6234	0.3163	0.3070	0.2764
LDA	0.6656	0.6433	0.6456	0.3118	0.2950	0.3014
AdaBoost	0.6524	0.6333	0.6190	0.3054	0.2890	0.2758
Bagging	0.6454	0.6380	0.6160	0.2912	0.2830	0.2682
GBM	0.6609	0.6470	0.6260	0.3106	0.3023	0.2822
Multi-layer Perceptron	0.6390	0.6229	0.6091	0.2921	0.2824	0.2731
XgBoost	0.6604	0.6450	0.6183	0.3133	0.2996	0.2736
Average	**0.6574**	**0.6446**	**0.6286**	**0.3054**	**0.2952**	**0.2820**

Table 9: Performance comparison of 12 different predictive models in Setting B (trained on synthetic, tested on real) in terms of AUROC and AUPRC (the generators of PATE-GAN and DPGAN are $(1, 10^{-5})$-differentially private). GAN is (∞, ∞)-differentially private and is given to indicate an upper bound of PATE-GAN and DPGAN.

UNOS HEART WAIT DATASET RESULT

	AUROC			AUPRC		
	GAN	**PATE-GAN**	**DPGAN**	GAN	**PATE-GAN**	**DPGAN**
Logistic Regression	0.6407	0.6155	0.5548	0.6691	0.6450	0.5901
Random Forests	0.6159	0.5950	0.5574	0.6436	0.6129	0.5768
Gaussian Naive Bayes	0.6323	0.6015	0.5343	0.6648	0.6371	0.5824
Bernoulli Naive Bayes	0.6213	0.6045	0.5763	0.6501	0.6363	0.6077
Linear SVM	0.6244	0.5979	0.5581	0.6486	0.6254	0.5892
Decision Tree	0.6209	0.6019	0.5590	0.6496	0.6284	0.5819
LDA	0.6403	0.6077	0.5530	0.6682	0.6406	0.5882
AdaBoost	0.6222	0.5928	0.5527	0.6404	0.6194	0.5826
Bagging	0.6084	0.5858	0.5493	0.6325	0.6074	0.5693
GBM	0.6374	0.6040	0.5585	0.6679	0.6352	0.5920
Multi-layer Perceptron	0.6328	0.5927	0.5562	0.6629	0.6240	0.5856
XgBoost	0.6362	0.5956	0.5533	0.6676	0.6267	0.5880
Average	**0.6277**	**0.5996**	**0.5552**	**0.6554**	**0.6282**	**0.5862**

Table 10: Performance comparison of 12 different predictive models in Setting B (trained on synthetic, tested on real) in terms of AUROC and AUPRC (the generators of PATE-GAN and DPGAN are $(1, 10^{-5})$-differentially private). GAN is (∞, ∞)-differentially private and is given to indicate an upper bound of PATE-GAN and DPGAN.

KAGGLE CERVICAL CANCER DATASET RESULT

	AUROC			AUPRC		
	GAN	**PATE-GAN**	**DPGAN**	GAN	**PATE-GAN**	**DPGAN**
Logistic Regression	0.9188	0.9102	0.8945	0.5949	0.5605	0.4672
Random Forests	0.9515	0.9373	0.9237	0.6366	0.6361	0.5735
Gaussian Naive Bayes	0.9393	0.8890	0.7973	0.5605	0.4422	0.3702
Bernoulli Naive Bayes	0.8421	0.8331	0.8296	0.2491	0.2211	0.2160
Linear SVM	0.9282	0.9086	0.9050	0.6031	0.5921	0.5665
Decision Tree	0.9451	0.9434	0.9283	0.6455	0.6094	0.5734
LDA	0.9358	0.9155	0.8667	0.6518	0.6061	0.5629
AdaBoost	0.9361	0.8898	0.7989	0.6881	0.5587	0.4281
Bagging	0.9425	0.9275	0.9080	0.6257	0.5871	0.5809
GBM	0.9398	0.9333	0.9017	0.6927	0.6165	0.5422
Multi-layer Perceptron	0.9005	0.9064	0.7933	0.5675	0.5246	0.3746
XgBoost	0.9408	0.9351	0.8919	0.6784	0.5978	0.5657
Average	**0.9268**	**0.9108**	**0.8699**	**0.5994**	**0.5460**	**0.4851**

Table 11: Performance comparison of 12 different predictive models in Setting B (trained on synthetic, tested on real) in terms of AUROC and AUPRC (the generators of PATE-GAN and DPGAN are $(1, 10^{-5})$-differentially private). GAN is (∞, ∞)-differentially private and is given to indicate an upper bound of PATE-GAN and DPGAN.

BLOCK DIAGRAMS

PATE-GAN

The two figures below indicate the iterative training procedure carried out by PATE-GAN; the figures correspond to a single generator update.

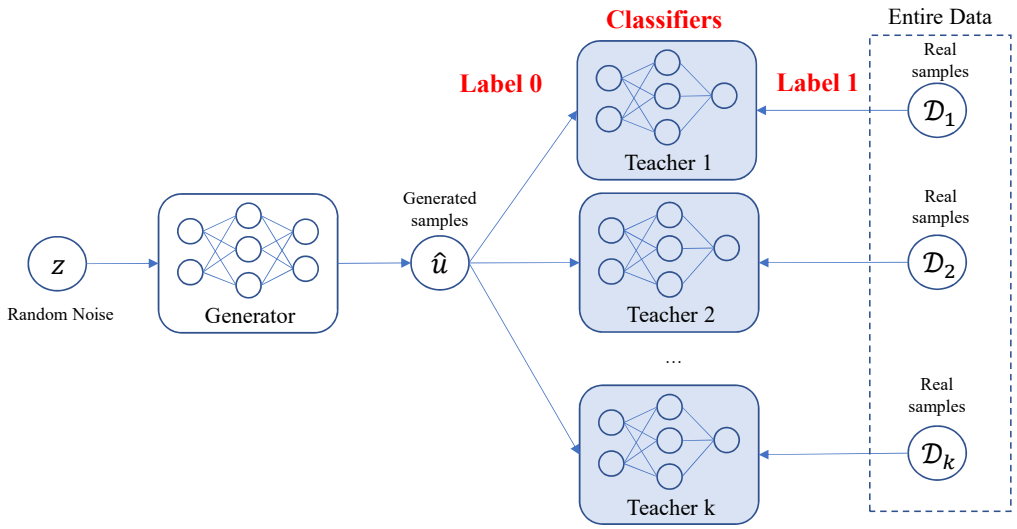


Figure 3: Block diagram of the training procedure for the teacher-discriminator during a single generator iteration. Teacher-discriminators are trained to minimize the classification loss when classifying samples as real samples or generated samples. During this step only the parameters of the teachers are updates (and not the generator).

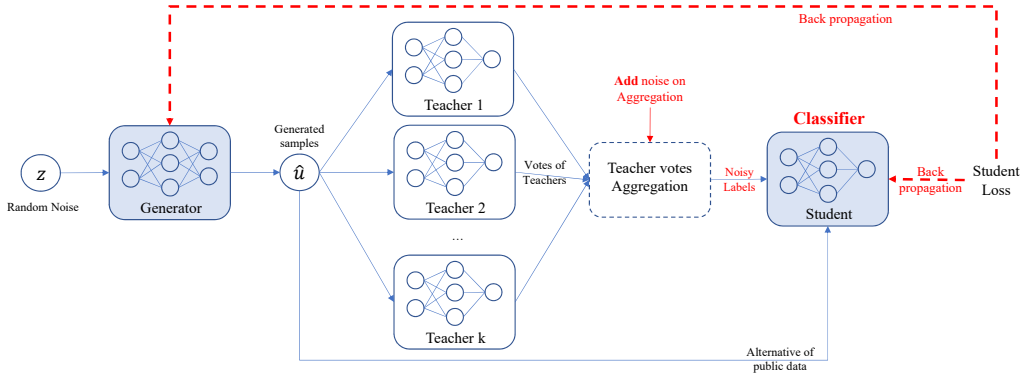


Figure 4: Block diagram of the training procedure for the student-discriminator and the generator. The student-discriminator is trained using noisy teacher-labelled generated samples (the noise provides the DP guarantees). The student is trained to minimize classification loss on this noisily labelled dataset, while the generator is trained to maximize the student loss. Note that the teachers are not updated during this step, only the student and the generator.

DPGAN [32]

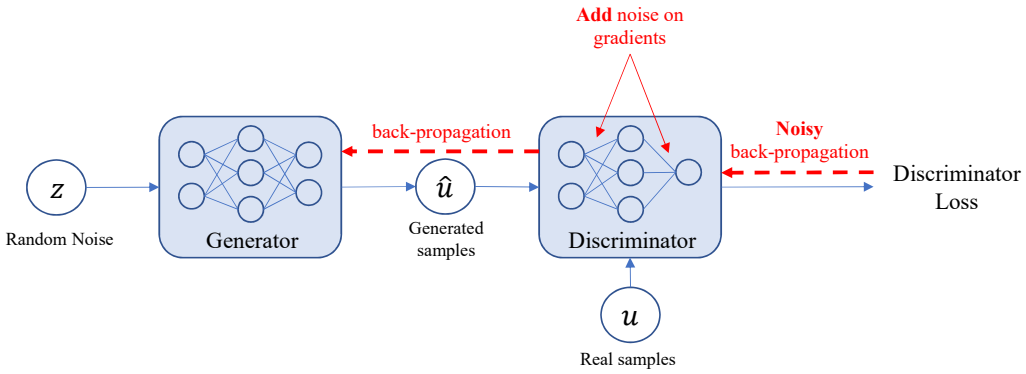


Figure 5: Block diagram of the DPGAN benchmark. It uses the standard WGAN framework. To guarantee differential privacy of the generator (with Post-processing Theorem), noise is added to the gradient of the discriminator during training to create a differentially private discriminator.

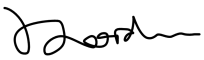
Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees
Publication Status	☑ Published □ Accepted for Publication □Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	James Jordon, Jinsung Yoon and Mihaela van der Schaar. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *International Conference on Learning Representations*, 2019.

Student Confirmation

Student Name:	James Jordon		
Contribution to the Paper	• The idea for the paper was mine. • I proposed the generator-driven knowledge transfer. • I wrote the entire paper with the exception of some parts of the related works sections and the initial basic draft of the experiments section, which Jinsung wrote. • I wrote Algorithm 1, which Jinsung then used to write the code for the paper. • Experimental design and analysis was done jointly. • Jinsung spent a lot of time tuning hyperparameters and investigating appropriate teacher discriminator architectures. • I proposed the Synthetic Ranking Agreement metric (which was published in a workshop paper). • All authors contributed to the development of the paper through multiple discussions. • All authors reviewed the final draft.		
Signature		Date	09/06/2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Professor Mihaela van der Schaar			
Supervisor comments I agree that the candidate has made a substantial contribution to the publication.			
Signature		Date	09/06/2021

This completed form should be included in the thesis, at the end of the relevant chapter.

Differentially Private Bagging: Improved utility and cheaper privacy than subsample-and-aggregate

James Jordon
University of Oxford
james.jordon@wolfson.ox.ac.uk

Jinsung Yoon
University of California, Los Angeles
jsyoon0823@g.ucla.edu

Mihaela van der Schaar
University of Cambridge
University of California, Los Angeles
Alan Turing Institute
mv472@cam.ac.uk, mihaela@ee.ucla.edu

Abstract

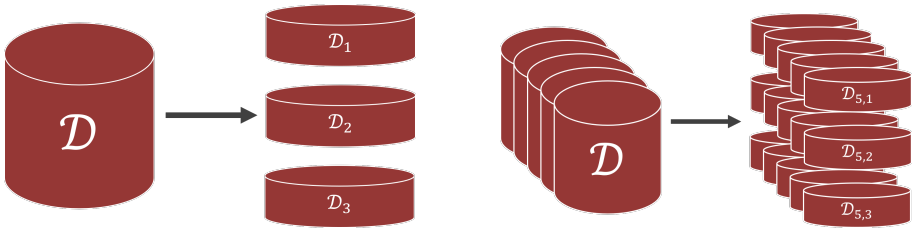
Differential Privacy is a popular and well-studied notion of privacy. In the era of big data that we are in, privacy concerns are becoming ever more prevalent and thus differential privacy is being turned to as one such solution. A popular method for ensuring differential privacy of a classifier is known as subsample-and-aggregate, in which the dataset is divided into distinct chunks and a model is learned on each chunk, after which it is aggregated. This approach allows for easy analysis of the model on the data and thus differential privacy can be easily applied. In this paper, we extend this approach by dividing the data several times (rather than just once) and learning models on each chunk within each division. The first benefit of this approach is the natural improvement of utility by aggregating models trained on a more diverse range of subsets of the data (as demonstrated by the well-known bagging technique). The second benefit is that, through analysis that we provide in the paper, we can derive tighter differential privacy guarantees when several queries are made to this mechanism. In order to derive these guarantees, we introduce the upwards and downwards moments accountants and derive bounds for these moments accountants in a data-driven fashion. We demonstrate the improvements our model makes over standard subsample-and-aggregate in two datasets (Heart Failure (private) and UCI Adult (public)).

1 Introduction

In the era of big data that we live in today, privacy concerns are becoming ever more prevalent. It falls to the researchers using the data to ensure that adequate measures are taken to ensure any results that are put into the public domain (such as the parameters of a model learned on the data) do not disclose sensitive attributes of the real data. For example, it is well known that the high capacity of deep neural networks can cause the networks to "memorize" training data; if such a network's parameters were made public, it may be possible to deduce some of the training data that was used to train the model, thus resulting in real data being leaked to the public.

Several attempts have been made at rigorously defining what it means for an algorithm, or an algorithm's output, to be "private". One particularly attractive and well-researched notion is that of differential privacy [1]. Differential privacy is a formal definition that requires that the distribution of the output of a (necessarily probabilistic) algorithm not be too different when a single data point is included in the dataset or not. Typical methods for enforcing differential privacy involve bounding

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.



(a) Subsample-and-aggregate (b) Differentially Private Bagging

Figure 1: A comparison of how the dataset is used in (a) subsample-and-aggregate and (b) our differentially private bagging procedure. By partitioning the dataset *multiple times* we are able to perform a tighter privacy analysis using our personalised moments accountant in addition to learning a better performing underlying classifier.

the effect that inclusion of a single sample can have on the output and then adding noise (typically Laplacian or Gaussian) proportional to this effect. The most difficult step in this process is in attaining a good bound on the effect of inclusion.

One method for bypassing this difficulty, is to build a classifier by dividing up the dataset into distinct subsets, training a separate classifier on each chunk, and then aggregating these classifiers. The effect of a single sample is then bounded by the fact that it was used only to train exactly one of these models and thus its inclusion or exclusion will affect only that model's output. By dividing the data into smaller chunks, we learn more models and thus the one model that a sample can effect becomes a smaller "fraction" of the overall model, thus resulting in a smaller effect that any one sample has on the model as a whole. This method is commonly referred to as subsample-and-aggregate [2, 3, 4].

In this work, we propose an extension to the subsample-and-aggregate methodology that has similarities with bagging [5]. Fig. 1 depicts the key methodological difference between standard subsample-and-aggregate and our proposed framework, Differentially Private Bagging (DPBag), namely that we partition the dataset many times. This multiple-partitioning not only improves utility by building a better predictor, but also enjoys stronger privacy guarantees due to the fact that the effect of adding or removing a single sample can be more tightly bounded within our framework. In order to prove these guarantees, we introduce the *personalised moments accountants*, which are data-driven variants of the moments accountant [6], that allow us to track the privacy loss with respect to each sample in the dataset and then deduce the final privacy loss by taking the maximum loss over all samples. The personalised moments accountant also lends itself to allowing for personalised differential privacy [7] in which we may wish to allow each individual to specify their own privacy parameters.

We demonstrate the efficacy of our model on two classification tasks, demonstrating that our model is an improvement over the standard subsample-and-aggregate algorithm.

2 Related Works

Several works have proposed methods for differentially private classification. Of particular interest is the method of [6], in which they propose a method for differentially private training of deep neural networks. In particular, they introduce a new piece of mathematical machinery, the moments accountant. The moments accountant allows for more efficient composition of differentially private mechanisms than either simple or advanced composition [1]. Fortunately, the moments accountant is not exclusive to deep networks and has proven to be useful in other works. In this paper, we use two variants of the moments accountant, which we refer to collectively as the personalised moments accountants. Our algorithm lends itself naturally to being able to derive tighter bounds on these personalised moments accountants than would be possible on the "global" moments accountant.

Most other methods use the subsample-and-aggregate framework (first discussed in [2]) to guarantee differential privacy. A popular, recent subsample-and-aggregate method is Private Aggregation of Teacher Ensembles (PATE), proposed in [8]. Their main contribution is to provide a data-driven bound on the moments accountant for a given query to the subsample-and-aggregate mechanism that

they claim significantly reduces the privacy cost over the standard data-independent bound. This is further built on in [9] by adding a mechanism that first determines whether or not a query will be too expensive to answer or not, only answering those that are sufficiently cheap. Both works use standard subsample-and-aggregate in which the data is partitioned only once. Our method is more fundamental than PATE, in the sense that the techniques used by PATE to improve on subsample-and-aggregate would also be applicable to our differentially private bagging algorithm. The bound they derive in [8] on the moments accountant should translate to our personalised moments accountants in the same way the data-independent bound does (i.e. by multiplying the dependence on the inverse noise scale by a data-driven value) and as such our method would provide privacy improvements over PATE similar to the improvements it provides over standard subsample-and-aggregate. We give an example of our conjectured result for PATE in the Supplementary Materials for clarity.

Another method that utilises subsample-and-aggregate is [10], in which they use the distance to instability framework [4] combined with subsample-and-aggregate to privately determine whether a query can be answered *without* adding any noise to it. In cases where the query can be answered, no privacy cost is incurred. Whenever the query cannot be answered, no answer is given but a privacy cost is incurred. Unfortunately, the gains to be had by applying our method over basic subsample-and-aggregate to their work are not clear, but we believe that at the very least, the utility of the answer provided may be improved on due to the ensemble having a higher utility in our case (and the same privacy guarantees will hold that they prove).

In [11], they build a method for learning a differentially private decision tree. Although they apply bagging to their framework, they do not do so to create privacy, but only to improve the utility of their learned classifier. The privacy analysis they provide is performed only on each individual tree and not on the ensemble as a whole.

3 Differential Privacy

Let us denote the feature space by $\mathcal{X}$, the set of possible class labels by $\mathcal{C}$ and write $\mathcal{U} = \mathcal{X} \times \mathcal{C}$. Let us denote by D the collection of all possible datasets consisting of points in $\mathcal{U}$. We will write $\mathcal{D}$ to denote a dataset in D, so that $\mathcal{D} = \{u_i\}_{i=1}^N = \{(x_i, y_i)\}_{i=1}^N$ for some N.

We first provide some preliminaries on differential privacy [1] before describing our method; we refer interested readers to [1] for a thorough exposition of differential privacy. We will denote an algorithm by $\mathcal{M}$, which takes as input a dataset $\mathcal{D}$ and outputs a value from some output space, $\mathcal{R}$.

Definition 1 (Neighboring Datasets [1])**.** Two datasets $\mathcal{D}, \mathcal{D}'$ are said to be neighboring if

$$\exists u \in \mathcal{U} \text{ s.t. } \mathcal{D} \setminus \{u\} = \mathcal{D}' \text{ or } \mathcal{D}' \setminus \{u\} = \mathcal{D}.$$

Definition 2 (Differential Privacy [1])**.** A *randomized* algorithm, $\mathcal{M}$, is (ϵ, δ)-differentially private if for all $\mathcal{S} \subset \mathcal{R}$ and for all neighboring datasets $\mathcal{D}, \mathcal{D}'$:

$$\mathbb{P}(\mathcal{M}(\mathcal{D}) \in \mathcal{S}) \leq e^{\epsilon}\mathbb{P}(\mathcal{M}(\mathcal{D}') \in \mathcal{S}) + \delta$$

where $\mathbb{P}$ is taken with respect to the randomness of $\mathcal{M}$.

Differential privacy provides an intuitively understandable notion of privacy - a particular sample's inclusion or exclusion in the dataset does not change the probability of a particular outcome very much: it does so by a multiplicative factor of e^{ϵ} and an additive amount, δ.

4 Differentially Private Bagging

In order to enforce differential privacy, we must bound the effect of a sample's inclusion or exclusion on the output of the model. In order to do this, we propose a model for which the maximal effect can be easily deduced and moreover, for which we can actually show a lesser maximal effect by analysing the training procedure and deriving data-driven privacy guarantees.

We begin by considering k (random) partitions of the dataset, $\mathcal{D}_1, ..., \mathcal{D}_k$ with $\mathcal{D}_i = \{D_1^i, ..., D_n^i\}$ for each i, where D_j^i is a set of size $\lfloor \frac{|\mathcal{D}|}{n} \rfloor$ or $\lceil \frac{|\mathcal{D}|}{n} \rceil$. We then train a "teacher" model, T_{ij} on each of these sets (i.e. T_{ij} is trained on D_j^i). We note that each sample $u \in \mathcal{D}$ is in precisely one set from each partition and thus in precisely k sets overall; it is therefore used to train k teachers. We

collect the indices of the corresponding teachers in the set $I(u) = \{(i,j) : u \in D_j^i\}$ and denote by $\mathcal{T}(u) = \{T_{ij} : (i,j) \in I(u)\}$ the set of teachers trained using the sample u.

Given a new sample to classify $x \in \mathcal{X}$, we first compute for each class the number of teachers that output that class, $n_c(x) = |\{(i,j) : T_{ij}(x) = c\}|$. The model then classifies the sample as

$$\hat{c}(x) = \arg\max\{n_c(x) : c \in \mathcal{C}\}$$

i.e. by classifying it as the class with the most votes. To make the output differentially private, we can add independent Laplacian noise to each of the resulting counts before taking $\arg\max$. So that the classification becomes

$$\tilde{c}_\lambda(x) = \arg\max\{n_c(x) + Y_c : c \in \mathcal{C}\}$$

where $Y_c, c \in \mathcal{C}$ are independent $Lap(\frac{k}{\lambda})$ random variables and where λ is a hyper-parameter of our model. We scale the noise to the number of partitions because the number of partitions is precisely the total number of teachers that any individual sample can effect. Thus the (naive) bound on the ℓ_1-sensitivity of this algorithm is k, giving us the following theorem, which tells us that our differentially private bagging algorithm is *at least* as private as the standard subsample-and-aggregate mechanism, independent of the number of partitions used.

Theorem 1. With k partitions and n teachers per partition, $\tilde{c}_\lambda$ is 2λ-differentially private with respect to the data $\mathcal{D}$.

Proof. This follows immediately from noting that the ℓ_1-sensitivity of $n_c(x)$ is k. See [1]. □

We note that the standard subsample-and-aggregate algorithm can be recovered from ours by setting $k = 1$. In the next section, we will derive tighter bounds on the differential privacy of our bagging algorithm when several queries are made to the classifier.

4.1 Personalised Moments Accountants

In order to provide tighter differential privacy guarantees for our method, we now introduce the personalised moments accountants. Like the original moments accountant from [6], these will allow us to compose a sequence of differentially private mechanisms more efficiently than using standard or advanced composition [1]. We begin with a preliminary definition (found in [6]).

Definition 3 (Privacy Loss and Privacy Loss Random Variable [6])**.** Let $\mathcal{M} : D \to \mathcal{R}$ be a randomized algorithm, with $\mathcal{D}$ and $\mathcal{D}'$ a pair of neighbouring datasets. Let aux be any auxiliary input. For any outcome $o \in \mathcal{R}$, we define the privacy loss at o to be:

$$c(o; \mathcal{M}, aux, \mathcal{D}, \mathcal{D}') = \log \frac{\mathbb{P}(\mathcal{M}(\mathcal{D}, aux) = o)}{\mathbb{P}(\mathcal{M}(\mathcal{D}', aux) = o)}$$

with the privacy loss *random variable*, C, being defined by

$$C(\mathcal{M}, aux, \mathcal{D}, \mathcal{D}') = c(\mathcal{M}(\mathcal{D}, aux), aux, \mathcal{D}, \mathcal{D}')$$

i.e. the random variable defined by evaluating the privacy loss at a sample from $\mathcal{M}(\mathcal{D}, aux)$.

In defining the moments accountant, an intermediate quantity, referred to by [6] as the "l-th moment" is introduced. We divide the definition of this l-th moment into a downwards and an upwards version (corresponding to whether $\mathcal{D}'$ is obtained by either removing or adding an element to $\mathcal{D}$, respectively). We do this because the upwards moments accountant must be bounded among *all* possible points $u \in \mathcal{U}$ that could be added, whereas the downwards moments accountants need only consider the points that are already in $\mathcal{D}$.

Definition 4. Let $\mathcal{D}$ be some dataset and let $u \in \mathcal{D}$. Let aux be any auxiliary input. Then the downwards moments accountant is given by

$$\check{\alpha}_\mathcal{M}(l; aux, \mathcal{D}, u) = \log \mathbb{E}(\exp(lC(\mathcal{M}, aux, \mathcal{D}, \mathcal{D} \setminus \{u\}))).$$

Definition 5. Let $\mathcal{D}$ be some dataset. Then the upwards moments accountant is defined as

$$\hat{\alpha}_\mathcal{M}(l; aux, \mathcal{D}) = \max_{u \in \mathcal{U}} \log \mathbb{E}(\exp(lC(\mathcal{M}, aux, \mathcal{D}, \mathcal{D} \cup \{u\}))).$$

We can recover the original moments accountant from [6], $\alpha_\mathcal{M}(l)$, as

$$\alpha_\mathcal{M}(l) = \max_{aux, \mathcal{D}} \{\hat{\alpha}_\mathcal{M}(l; aux, \mathcal{D}), \max_u \check{\alpha}_\mathcal{M}(l; aux, \mathcal{D}, u)\}. \tag{1}$$

We will use this fact, together with the two theorems in the following subsection, to calculate the final global privacy loss of our mechanism.

4.2 Results inherited from the Moments Accountant

The following two theorems state two properties that our personalised moments accountants share with the original moments accountant. Note that the composability in Theorem 2 is being applied to *each* personalised moments accountant individually.

Theorem 2 (Composability)**.** Suppose that an algorithm $\mathcal{M}$ consists of a sequence of adaptive algorithms (i.e. algorithms that take as auxiliary input the outputs of the previous algorithms) $\mathcal{M}_1, ..., \mathcal{M}_m$ where $\mathcal{M}_i : \prod_{j=1}^{i-1} \mathcal{R}_j \times D \to \mathcal{R}_i$. Then, for any l

$$\check{\alpha}_{\mathcal{M}}(l; \mathcal{D}, u) \leq \sum_{i=1}^{m} \check{\alpha}_{\mathcal{M}_i}(l; \mathcal{D}, u)$$

and

$$\hat{\alpha}_{\mathcal{M}}(l; \mathcal{D}) \leq \sum_{i=1}^{m} \hat{\alpha}_{\mathcal{M}_i}(l; \mathcal{D}).$$

Proof. The statement of this theorem is a variation on Theorem 2 from [6], applied to the personalised moments accountants. Their proof involves proving this stronger result. See [6], Theorem 2 proof. □

Theorem 3 ((ϵ, δ) from $\alpha(l)$ [6])**.** Let $\delta > 0$. Any mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private for

$$\epsilon = \min_{l} \frac{\alpha_{\mathcal{M}}(l) + \log(\frac{1}{\delta})}{l} \tag{2}$$

Proof. See [6], Theorem 2. □

Theorem 2 means that bounding each personalised moments accountant individually could provide a significant improvement on the overall bound for the moments accountant. Combined with Eq. 1, we can first sum over successive steps of the algorithm *and then* take the maximum. In contrast, original approaches that bound only the overall moments accountant at each step essentially compute

$$\alpha_{\mathcal{M}}(l) = \sum_{i=1}^{m} \max_{aux, \mathcal{D}} \{\hat{\alpha}_{\mathcal{M}_i}(l; aux, \mathcal{D}), \max_{u} \check{\alpha}_{\mathcal{M}_i}(l; aux, \mathcal{D}, u)\}. \tag{3}$$

Our approach of bounding the personalised moments accountant allows us to compute the bound as

$$\alpha_{\mathcal{M}}(l) = \max_{aux, \mathcal{D}} \{\sum_{i=1}^{m} \hat{\alpha}_{\mathcal{M}_i}(l; aux, \mathcal{D}), \max_{u} \sum_{i=1}^{m} \check{\alpha}_{\mathcal{M}_i}(l; aux, \mathcal{D}, u)\} \tag{4}$$

which is strictly smaller whenever there is not some personalised moments accountant that is *always* larger than *all* other personalised moments accountants. The bounds we derive in the following subsection and the subsequent remarks will make clear why this is an unlikely scenario.

4.3 Bounding the Personalised Moments Accountants

Having defined the personalised moments accountants, we can now state our main theorems, which provide a data-dependent bound on the personalised moments accountant for a single query to $\tilde{c}_\lambda$.

Theorem 4 (Downwards bound)**.** Let $x_{new} \in \mathcal{X}$ be a new point to classify. For each $c \in \mathcal{C}$ and each $u \in \mathcal{D}$, define the quantities

$$n_c(x_{new}; u) = \frac{|\{(i, j) \in I(u) : T_{ij}(x_{new}) = c\}|}{k}$$

i.e. $n_c(x_{new}; u)$ is the fraction of teachers *that were trained on a dataset containing* u that output class c when classifying x_{new}. Let

$$m(x_{new}; u) = \max_{c} \{1 - n_c(x_{new}; u)\}.$$

Then

$$\check{\alpha}_{\tilde{c}_\lambda(x_{new})}(l; \mathcal{D}, u) \leq 2\lambda^2 m(x_{new}; u)^2 l(l + 1). \tag{5}$$

Proof. (Sketch.) The theorem follows from the fact that $m(x_{new}; u)$ is the maximum change that can occur in the vote fractions, $n_c, c \in \mathcal{C}$ when the sample u is removed from the training of each model in $\mathcal{T}(u)$, corresponding to all teachers that were not already voting for the minority class switching their vote to the minority class. m can thus be thought of as the personalised ℓ_1-sensitivity of a specific query to our algorithm, and so the standard sensitivity based argument gives us that $\tilde{c}_\lambda(x_{new})$ is $2\lambda m(x_{new}; u)$-differentially private *with respect to removing* u. The bound on the (downwards) moments accountant then follows using a similar argument to the proof of Prop. 3.3 in [12]. □

To prove the upwards bound, we must understand what happens when we add a point to our training data - which is that it will be added to a training set for precisely 1 teacher in each of the k partitions. Each dataset in a partition will either be of size $\lceil \frac{|\mathcal{D}|}{n} \rceil$ or $\lfloor \frac{|\mathcal{D}|}{n} \rfloor$. We assume (without loss of generality) that a new point is added to the first dataset in each partition that contains $\lfloor \frac{|\mathcal{D}|}{n} \rfloor$ samples. We collect the indices of these datasets in $I(*)$ and denote the set of teachers trained on these subsets by $\mathcal{T}(*)$.

Theorem 5 (Upwards bound)**.** Let $x_{new} \in \mathcal{X}$ be a new point to classify. For each $c \in \mathcal{C}$, define the quantity

$$n_c(x_{new}; *) = \frac{|\{(i,j) \in I(*) : T_{ij}(x_{new}) = c\}|}{k}$$

i.e. $n_c(x_{new}; *)$ is the fraction of teachers *whose training set would receive the new point* that output class c when classifying x_{new}. Let

$$m(x_{new}; *) = \max_c \{1 - n_c(x_{new}; *)\}.$$

Then

$$\hat{\alpha}_{\tilde{c}_\lambda(x_{new})}(l; \mathcal{D}) \leq 2\lambda^2 m(x_{new}; *)^2 l(l+1). \tag{6}$$

Proof. The proof is exactly as for Theorem 4, replacing $I(u)$ and $\mathcal{T}(u)$ with $I(*)$ and $\mathcal{T}(*)$. □

The standard bound on the moments accountant of a 2λ differentially private algorithm is $2\lambda^2 l(l+1)$ (see [12]). Thus, our theorems introduce a factor of $m(x_{new}; u)^2$. Note that by definition $m \leq 1$ and thus our bound is in general tighter, but always at least as tight. It should be noted, however, that for a single query, this bound may not improve on the naive $2\lambda^2 l(l+1)$ bound, since in that case equations 3 and 4 are equal. If there is *any* training sample $u \in \mathcal{D} \cup \{*\}$ and any class $c \in \mathcal{C}$ for which *all* teachers in $\mathcal{T}(u)$ classify x_{new} as some class other than c then $m(x_{new}; u) = 1$. However, over the course of several queries, it is unlikely that each set of teachers $\mathcal{T}(u)$ always exclude some class, and as such the total bound according to Theorems 2, 4 and 5 is lower than if we just used the naive bound. In the case of binary classification, for example, the bounds are only the same if there is some set of teachers that are *always* unanimous when classifying new samples.

Remarks. (i) $m(x_{new}; u)$ is smallest when the teachers in $\mathcal{T}(u)$ are divided evenly among the classes when classifying x_{new}, this is intuitive because in such a situation, u is providing very little information about how to classify u and thus little is being leaked about u when we classify x_{new}.
(ii) $m(u)$ is bounded below by $1 - \frac{1}{|\mathcal{C}|}$ and so our method will provide the biggest improvements for binary classification and the improvements will decay as the number of classes increases.
(iii) When $k = 1$, $m(u)$ is always 1 because n_c is 1 for some $c \in \mathcal{C}$ and then 0 for all remaining classes and from this we recover the standard bound of $2\lambda l(l+1)$ used for subsample-and-aggregate.
(iv) For Eq. 3 and 4 to be equal, there must exist some u^* for which $m(x_{new}; u^*) > m(x_{new}; u)$ for *all* u and x_{new}. This amounts to there being some set of teachers (corresponding to u^*) that are in more agreement than *every* other set of teachers for *every* new point they are asked to classify. Other than in this unlikely scenario, Eq. 4 will be *strictly* smaller than Eq. 3.

4.4 Semi-supervised knowledge transfer

We now discuss how best to leverage the fact that the best gains from our approach come from answering several queries (as implied by equations 3 and 4). We first note that the vanilla subsample-and-aggregate method does not derive data-dependent privacy guarantees for an individual query, and thus, for a fixed ϵ and δ, the number of queries that can be answered by the mechanism is known in advance. In contrast, because our data-driven bounds on the personalised moments accountants

depend on the queries themselves, the cost of any given query is not known in advance and as such the number of queries we *can* answer before using up our privacy allowance (ϵ) is unknown.

Unfortunately, we cannot simply answer queries until the allowance is used up, because the number of queries that we answer is a function of the data itself and thus we would need to introduce a differentially private mechanism for determining when to stop (such as calculating ϵ and δ after each query using smooth-sensitivity as proposed in [8]). Instead, we follow [8] and leverage the fact that we can answer more queries than standard subsample-and-aggregate to train a *student* model using unlabelled public data. The final output of our algorithm will then be a trained classifier that can be queried indefinitely. To train this model, we take unlabelled public data $\mathcal{P} = \{\tilde{x}_1, \tilde{x}_2, ...\}$ and label it using $\tilde{c}_\lambda$ until the privacy allowance has been used up. This will result in a (privately) labelled dataset $\tilde{\mathcal{P}} = \{(\tilde{x}_1, y_1), ..., (\tilde{x}_p, y_p)\}$ where p is the number of queries answered. We train a student model, S, on this dataset and the resultant classifier can now be used to answer any future queries. Because of our data-driven bound on the personalised moments accountant, we will typically have that $p > q$ where q is the number of queries that can be answered by a standard subsample-and-aggregate procedure. The pseudo-code for learning a differentially private student classifier using our differentially private bagging model is given in Algorithm 1 (pseudo-code for training a student model using standard subsample-and-aggregate is given in the Supplementary Materials for comparison). Note that the majority of for loops (in particular the one on line 18) can be parallelized.

Algorithm 1 Semi-supervised differentially private knowledge transfer using multiple partitions

1: **Input:** ϵ, δ, $\mathcal{D}$, batch size n_{mb}, number of partitions k, number of teachers per partition n, noise size λ, maximum order of moments to be explored, L, unlabelled public data $\mathcal{D}_{pub}$
2: **Initialize:** $\{\theta_T^{i,j}\}_{i=1,j=1}^{k,n}$, θ_S, $\hat{\epsilon} = 0$, $\alpha(l; x) = 0$ for $l = 1, ..., L$, $x \in \mathcal{D} \cup \{*\}$
3: Create n partitions of the dataset which are each made up of n disjoint subsets of the data $\mathcal{D}_{i,j}$, $i = 1, ..., n$, $j = 1, ..., k$ such that $\bigcup_i \mathcal{D}_{i,j} = \mathcal{D}$ and $\mathcal{D}_{i_1,j} \cap \mathcal{D}_{i_2,j} = \emptyset$ for all $i_1 \neq i_2, j$
4: Set $I(*) = \{(n, 1), ..., (n, k)\}$
5: **while** Teachers have not converged **do**
6: **for** $i = 1, ..., n$ **do**
7: **for** $j = 1, ..., k$ **do**
8: Sample $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_{n_{mb}}, y_{n_{mb}}) \overset{\text{i.i.d.}}{\sim} D_{i,j}$
9: Update teacher, $T_{i,j}$, using SGD
10: $\nabla_{\theta_T^{i,j}} - \left[\sum_{s=1}^{n_{mb}} \sum_{c \in \mathcal{C}} y_{s,c} \log(T_{i,j}^c(\mathbf{x}_s))\right]$ (multi-task cross-entropy loss)
11: **while** $\hat{\epsilon} < \epsilon$ **do**
12: Sample $\mathbf{x}_1, ..., \mathbf{x}_{n_{mb}} \sim \mathcal{D}_{pub}$
13: **for** $s = 1, ..., n_{mb}$ **do**
14: $r_s \leftarrow \tilde{c}_\lambda(\mathbf{x}_s)$
15: Update the element-wise moments accountants
16: $n_c \leftarrow \frac{|\{(i,j):T_{i,j}(\mathbf{x}_s)=c\}|}{k}$ for $c \in \mathcal{C}$
17: **for** $x \in \mathcal{D} \cup \{*\}$ **do**
18: $n_c(x) \leftarrow \frac{|\{(i,j) \in I(x):T_{i,j}(\mathbf{x}_s)=c\}|}{k}$ for $c \in \mathcal{C}, x \in \mathcal{D}$
19: $m(x) \leftarrow \max_c\{1 - n_c(x)\}$
20: **for** $l = 1, ..., L$ **do**
21: $\alpha(l; x) \leftarrow \alpha(l; x) + 2\lambda^2 m(x)^2 l(l + 1)$
22: Update the student, S, using SGD
23: $\nabla_{\theta_S} - \sum_{s=1}^{n_{mb}} \sum_{c \in \mathcal{C}} r_{s,c} \log S^c(\mathbf{x}_s)$ (multi-task cross-entropy loss)
24: $\hat{\epsilon} \leftarrow \min_l \left[\max_x \frac{\alpha(l;x)+\log(\frac{1}{\delta})}{l}\right]$
25: **Output:** S

Theorem 6. The output of Algorithm 1 is (ϵ, δ)-differentially private with respect to $\mathcal{D}$.

Proof. This follows from Theorems 2, 3, 4 and 5. □

5 Experiments

In this section we compare our method (DPBag) against the standard subsample-and-aggregate framework (SAA) to illustrate the improvements that can be achieved at a fundamental level by using our model. Additionally, we compare against our method *without* the improved privacy bound (DPBAG-) to quantify the improvements that are due to the bagging procedure and those that are due to our improved privacy bound. We perform the experiments on two real-world datasets: Heart Failure and UCI Adult (dataset description and results for UCI Adult can be found in the Supplementary Materials). Implementation of DPBag can be found at `https://bitbucket.org/mvdschaar/mlforhealthlabpub/src/master/alg/dpbag/`.

Heart Failure dataset: The Heart Failure dataset is a private dataset consisting of 24175 patients who have suffered heart failure. We set the label of each patient as 3-year all-cause mortality, excluding all patients who are censored before 3 years. The total number of features is 29 and the number of patients is 24175. Among 24175 patients, 10387 patients (43.0%) die within 3 years.

We randomly divide the data into 3 disjoint subsets: (1) a training set (33%), (2) public data (33%), (3) a testing set (33%). In the main paper, we use logistic regression for the teacher and student models in both algorithms; additional results for Gradient Boosting Method (GBM) can be found in the Supplementary Materials. We set $\delta = 10^{-5}$. We vary $\epsilon \in \{1, 3, 5\}$, $n \in \{50, 100, 250\}$ and $k \in \{10, 50, 100\}$. In all cases we set $\lambda = \frac{2}{n}$. To save space, we report DPBag results for $n \in \{100, 250\}, k \in \{50, 100\}$ and SAA results for $n = 250$ (the best performing) in the main manuscript, with full tables reported in the Supplementary Materials. Results reported are the mean of 10 runs of each experiment.

5.1 Results

In Table 1 we report the accuracy, AUROC and AUPRC of the 3 methods and we also report these for a non-privately trained baseline model (NPB), allowing us to quantify how much has been "lost due to privacy". In Table 2, we report the total number of queries that could be made to each differentially private classifier before the privacy budget was used up.

In Table 1 we see that DPBag outperforms standard SAA for all values of ϵ with Table 2 showing that our method allows for a significant increase in the number of public samples that can be labelled (almost 100% more for $\epsilon = 3$).

The optimal number of teachers, n, varies with ϵ, for both DPBag and SAA. We see that for $\epsilon = 1$, $n = 250$ performs best, but as we increase ϵ the optimal number of teachers decreases. For small ϵ and small n, very few public samples can be labelled and so the student does not have enough data to learn from. On the other hand, for large ϵ and large n, the number of answered queries is much larger, to the point where now the limiting factor is not the *number* of labels but is instead the *quality* of the labels. Since we scale the noise to the number of teachers, the label quality improves with fewer teachers because each teacher is trained on a larger portion of the training data. This is reflected by both DPBag and SAA. In the SAA results, the performance does not saturate as quickly with respect to ϵ because the number of queries that ϵ corresponds to for SAA is smaller than for DPBag.

As expected, we see that DPBAG- sits between SAA and DPBAG, enjoying performance gains due to a stronger underlying model, and thus more accurately labelled training samples for the student, but the improved privacy bound that DPBAG allows more samples to be labelled and thus further gains are still made.

Table 2 also sheds light on the behavior of DPBag with repsect to k. We see in Table 1 that both $k = 50$ and $k = 100$ can provide the best performance (depending on n and ϵ). In Table 2, the number of queries that can be answered increases with k. This implies that (as expected), as we increase k, the quantity $m(u)$ gets closer to 0.5, and so each query costs less. However, when $m(u)$ is close to 0.5 for *all* samples, u, in the dataset then neither class will have a clear majority and thus the labels are more susceptible to flipping due to the noise added. $k = 50$ appears to balance this trade-off when ϵ is larger (and so we can already answer more queries) and when ϵ is smaller we see that answering more queries is more important than answering them well, so $k = 100$ is preferred.

Table 1: Prediction performance (Accuracy, AUROC, AUPRC) of DPBag and SAA with $\delta = 10^{-5}$ on the Heart Failure dataset using Logistic Regression. **Bold** indicates the best performance achieved for the given metric and fixed ϵ. DPBAG- is our method without the improved privacy analysis. NPB is a non-private baseline model, included to indicate an upper bound on our performance.

Model	n	k	Accuracy			AUROC			AUPRC		
			$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
DPBag	100	50	.5639	**.6085**	**.6154**	.5547	.6326	**.6453**	.4793	.5530	**.5656**
		100	.5667	.6050	.6142	.5626	.6295	.6448	.4895	.5496	.5652
	250	50	.5888	.6061	.6099	.5954	.6320	.6391	.5161	.5526	.5607
		100	**.5986**	.6077	.6091	**.6096**	**.6373**	.6398	**.5289**	**.5542**	.5644
DPBag-	100	50	.5614	.6019	.6128	.5544	.6288	.6429	.4792	.5411	.5612
		100	.5596	.6007	.6108	.5609	.6174	.6354	.4767	.5338	.5525
	250	50	.5855	.6051	.6086	.5896	.6295	.6366	.5093	.5498	.5565
		100	.5875	.6061	.6110	.5884	.6321	.6407	.5103	.5518	.5615
SAA	250		.5798	.6019	.6024	.5778	.6284	.6356	.5023	.5496	.5559
NPB	1	-		.6527			.6992			.6281	

Table 2: Number of labels provided by each method before the privacy budget, ϵ, is used up on the Heart Failure dataset. Note that DPBAG- and SAA have the same, non-data dependent privacy analysis and so provide the same number of labels as each other.

Models	n	k	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
DPBag	100	50	74	593	1487
		100	76	609	1538
	250	50	468	3785	6380
		100	507	4044	6805
SAA	250	-	264	2108	5269

6 Discussion

In this work, we introduced a new methodology for developing a differentially private classifier. Building on the ideas of subsample-and-aggregate, we divide the dataset *several* times, allowing us to derive (tighter) data-dependent bounds on the privacy cost of a query to our mechanism. To do so, we defined the personalised moments accountants, which we use to accumulate the privacy loss of a query with respect to each sample in the dataset (and any potentially added sample) individually.

A key advantage of our model, like subsample-and-aggregate is that it is model agnostic, and can be applied using any base learner, with the differential privacy guarantees holding regardless of the learner used.

We believe this work opens up several interesting avenues for future research: (i) the privacy guarantees could potentially be improved on by making assumptions about the base learners used, (ii) the personalised moments accountants naturally allow for the development of an algorithm that affords each sample a different level of differential privacy, i.e. personalised differential privacy [7], (iii) we believe bounds such as those derived in [8] and [9] that rely on the subsample-and-aggregate method will have natural analogs with respect to our bagging procedure corresponding to tighter bounds on the personalised moments accountants than can be shown for the global moments accountant using simple subsample-and-aggregate (see the discussion in the Supplementary Materials).

Acknowledgments

This work was supported by the National Science Foundation (NSF grants 1462245 and 1533983), and the US Office of Naval Research (ONR).

References

[1] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[2] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84. ACM, 2007.

[3] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *STOC*, volume 9, pages 371–380, 2009.

[4] Abhradeep Guha Thakurta and Adam Smith. Differentially private feature selection via stability arguments, and the robustness of the lasso. In *Conference on Learning Theory*, pages 819–850, 2013.

[5] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[6] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[7] Zach Jorgensen, Ting Yu, and Graham Cormode. Conservative or liberal? personalized differential privacy. In *2015 IEEE 31St international conference on data engineering*, pages 1023–1034. IEEE, 2015.

[8] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[9] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[10] Raef Bassily, Om Thakkar, and Abhradeep Thakurta. Model-agnostic private learning via stability. *arXiv preprint arXiv:1803.05101*, 2018.

[11] Xiaoqian Liu, Qianmu Li, Tao Li, and Dong Chen. Differentially private classification with decision tree ensemble. *Applied Soft Computing*, 62:807–816, 2018.

[12] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

Supplementary Materials
Differentially Private Bagging: Improved utility and cheaper privacy than subsample-and-aggregate

James Jordon
University of Oxford
james.jordon@wolfson.ox.ac.uk

Jinsung Yoon
University of California, Los Angeles
jsyoon0823@g.ucla.edu

Mihaela van der Schaar
University of Cambridge
University of California, Los Angeles
Alan Turing Institute
mv472@cam.ac.uk, mihaela@ee.ucla.edu

1 Discussion continued

As alluded to in the main paper, we believe that our work is a fundamental improvement over the subsample-and-aggregate model and as such can be used to improve (most) methods that build on the subsample-and-aggregate framework. To illustrate how we think this might look, we take Private Aggregation of Teacher Ensembles (PATE) [1] as an example (note that we do not prove the following result - it is a conjecture).

PATE builds on the standard subsample-and-aggregate model by deriving a data-dependent bound on the (global) moments accountant at each step of the form

$$\alpha(l) \leq \log((1-q)\left(\frac{1-q}{1-e^{2\lambda}q}\right)^{l} + qe^{2\lambda l}) \tag{1}$$

where

$$q = \sum_{c \neq c^*} \frac{2 + \lambda|n_c^* - n_c|}{4\exp(\lambda|n_c^* - n_c|)} \tag{2}$$

with n_c^* denoting the number of teachers voting for the majority class.

What we believe our algorithm offers here is to improve on this data-dependent bound in a similar fashion to the improvement we make in the paper, which amounts to replacing λ with $\lambda \times m$ and instead bounding the *personalised* moments accountants, so that

$$\check{\alpha}_{\check{c}_\lambda(x_{new})}(l; \mathcal{D}, u) \leq \log((1-q)\left(\frac{1-q}{1-e^{2\lambda m(x_{new};u)}q}\right)^{l} + qe^{2\lambda m(x_{new};u)l}). \tag{3}$$

While we are confident in this bound, it is unclear whether we can also replace the λ term in the definition of q, which we leave as an open question for future research.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

2 Full tables for the results on the Heart Failure dataset

2.1 Logistic Regression

Table 1: Prediction performance (Accuracy, AUROC, AUPRC) of DPBag and SAA with $\delta = 10^{-5}$ on the Heart Failure dataset using Logistic Regression. **Bold** indicates the best performance achieved for the given metric and fixed ϵ. NPB is a non-private baseline model, included to indicate an upper bound on our performance.

Model	n	k	Accuracy			AUROC			AUPRC		
			$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
DPBag	50	10	.5538	.5933	.6073	.5478	.6015	.6310	.4738	.5187	.5459
		50	.5628	.5941	.6123	.5516	.6134	.6385	.4760	.5317	.5560
		100	.5478	.5886	.6083	.5475	.6037	.6333	.4739	.5255	.5520
	100	10	.5635	.6051	.6137	.5581	.6285	.6431	.4828	.5467	.5625
		50	.5639	**.6085**	**.6154**	.5547	.6326	**.6453**	.4793	.5530	**.5656**
		100	.5667	.6050	.6142	.5626	.6295	.6448	.4895	.5496	.5652
	250	10	.5919	.6032	.6066	.6008	.6330	.6399	.5234	**.5542**	.5623
		50	.5888	.6061	.6099	.5954	.6320	.6391	.5161	.5526	.5607
		100	**.5986**	.6077	.6091	**.6096**	**.6373**	.6398	**.5289**	**.5542**	.5644
DPBag-	100	50	.5614	.6019	.6128	.5544	.6288	.6429	.4792	.5411	.5612
		100	.5596	.6007	.6108	.5609	.6174	.6354	.4767	.5338	.5525
	250	50	.5855	.6051	.6086	.5896	.6295	.6366	.5093	.5498	.5565
		100	.5875	.6061	.6110	.5884	.6321	.6407	.5103	.5518	.5615
SAA	50	-	.5468	.5842	.6002	.5367	.5927	.6159	.4596	.5086	.5310
	100		.5613	.6006	.6077	.5567	.6144	.6280	.4837	.5342	.5471
	250		.5798	.6019	.6024	.5778	.6284	.6356	.5023	.5496	.5559
NPB	1	-		.6527			.6992			.6281	

Table 2: Number of labels provided by each method using Logistic Regression before the privacy budget, ϵ, is used up on the Heart Failure dataset with $\delta = 10^{-5}$. Note that DPBAG- and SAA have the same, data-independent privacy analysis and so provide the same number of labels as each other.

Models	n	k	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
DPBag	50	10	17	120	301
		50	20	137	345
		100	20	139	351
	100	10	62	507	1277
		50	74	593	1487
		100	76	609	1538
	250	10	372	2978	7452
		50	468	3785	6380
		100	507	4044	6805
SAA	50	-	11	85	211
	100		43	338	843
	250		264	2108	5269

2.2 GBM

Table 3: Prediction performance (Accuracy, AUROC, AUPRC) of DPBag and SAA with $\delta = 10^{-5}$ on the Heart Failure dataset using GBM. **Bold** indicates the best performance achieved for the given metric and fixed ϵ. DPBAG- is our method without the improved privacy analysis. NPB is a non-private baseline model, included to indicate an upper bound on our performance.

Model	n	k	Accuracy			AUROC			AUPRC		
			$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
DPBag	100	50	.5740	.6002	.6121	.5827	.6239	.6307	.5073	.5473	.5641
		100	.5906	**.6165**	**.6239**	.5908	.6265	.6444	.5115	.5502	.5613
	150	50	**.5912**	.6060	.6164	**.5987**	.6281	.6322	**.5182**	.5423	.5656
		100	.5835	.6101	.6218	.5936	**.6289**	**.6451**	.5170	**.5504**	**.5691**
DPBag-	100	50	.5633	.5978	.6054	.5723	.6189	.6231	.4887	.5387	.5424
		100	.5665	.6061	.6136	.5714	.6203	.6334	.4894	.5393	.5521
	150	50	.5786	.6022	.6113	.5911	.6195	.6292	.5066	.5377	.5556
		100	.5751	.6014	.6186	.5864	.6193	.6355	.5158	.5433	.5523
SAA	100	-	.5763	.5959	.6068	.5796	.6132	.6261	.4948	.5284	.5457
	150		.5731	.5977	.6111	.5839	.6137	.6276	.5005	.5353	.5511
NPB	1	-		.6482			.6945			.6215	

Table 4: Number of labels provided by each method using GBM before the privacy budget, ϵ, is used up on the Heart Failure dataset with $\delta = 10^{-5}$. Note that DPBAG- and SAA have the same, data-independent privacy analysis and so provide the same number of labels as each other.

Models	n	k	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
DPBag	100	50	73	615	1527
		100	79	635	1591
	150	50	161	1332	3329
		100	174	1386	3473
SAA	100	-	43	338	843
	150		95	759	1897

3 Results on UCI Adult Dataset

UCI Adult dataset: UCI Adult dataset `https://archive.ics.uci.edu/ml/datasets/adult` is a public dataset for binary classification. The total number of features is 108 (after one-hot encoding) and the number of samples is 48841. Among 48841 samples, 11687 samples (23.9%) have class 1.

3.1 Logistic Regression

Table 5: Prediction performance (Accuracy, AUROC, AUPRC) of DPBag and SAA with $\delta = 10^{-5}$ on the UCI Adult Dataset using Logistic Regression. **Bold** indicates the best performance achieved for the given metric and fixed ϵ. NPB is a non-private baseline model, included to indicate an upper bound on our performance.

Model	n	k	Accuracy			AUROC			AUPRC		
			$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
DPBag	50	10	.7532	.7934	.8042	.7005	.8069	.8176	.4031	.5505	.5751
		50	.7384	.7908	.8056	.6816	.7714	.8187	.3837	.4988	.5690
		100	.7467	.7872	.8018	.6609	.7811	.8160	.3990	.5159	.5717
	100	10	.7721	.8061	.8221	.7386	.8005	.8237	.4514	.5503	.6068
		50	.7736	.8089	.8201	.7382	.8053	.8299	.4558	.5625	.6031
		100	.7707	.8092	.8233	.7379	.8051	.8388	.4497	.5648	.6216
	250	10	.7950	.8170	.8213	**.8062**	.8339	.8445	.5384	.6091	.6225
		50	.7938	.8199	**.8249**	.7831	**.8448**	**.8524**	.5216	**.6191**	**.6389**
		100	**.8035**	**.8207**	.8236	.7920	.8393	.8505	**.5433**	.6137	.6350
DPBag-	100	50	.7644	.7922	.8117	.7426	.7963	.8253	.4548	.5406	.5977
		100	.7697	.7989	.8130	.7490	.8002	.8291	.4631	.5503	.5985
	250	50	.7918	.8155	.8206	.7907	.8355	.8482	.5287	.6012	.6274
		100	.7919	.8156	.8220	.7922	.8301	.8466	.5259	.5993	.6291
SAA	50	-	.7107	.7766	.7939	.6553	.7701	.7934	.3755	.5072	.5485
	100		.7579	.7986	.8149	.7652	.8057	.8291	.4795	.5590	.6147
	250		.7883	.8103	.8137	.7825	.8261	.8378	.5193	.5976	.6163
NPB	1	-		.8472			.9029			.7526	

Table 6: Number of labels provided by each method using Logistic Regression before the privacy budget, ϵ, is used up on the UCI Adult Dataset with $\delta = 10^{-5}$. Note that DPBAG- and SAA have the same, data-independent privacy analysis and so provide the same number of labels as each other.

Models	n	k	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
DPBag	50	10	20	91	228
		50	21	94	235
		100	22	94	236
	100	10	52	372	938
		50	54	385	963
		100	55	390	973
	250	10	297	2371	5917
		50	310	2496	6240
		100	317	2533	6327
SAA	50	-	11	85	211
	100		43	338	843
	250		264	2108	5269

3.2 GBM

Table 7: Prediction performance (Accuracy, AUROC, AUPRC) of DPBag and SAA with $\delta = 10^{-5}$ on the UCI Adult Dataset using GBM. **Bold** indicates the best performance achieved for the given metric and fixed ϵ. DPBAG- is our method without the improved privacy analysis. NPB is a non-private baseline model, included to indicate an upper bound on our performance.

Model	n	k	Accuracy			AUROC			AUPRC		
			$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
DPBag	100	50	.7743	**.7986**	.8069	.7655	.7861	.7997	.4908	.5548	.5746
		100	**.7763**	.7947	.8028	.7406	.7820	.7879	.4641	.5505	.5648
	150	50	.7697	.7974	.8157	.7548	**.7873**	.8033	.4984	**.5581**	.5873
		100	.7730	.7963	**.8183**	.7582	.7802	**.8039**	**.5084**	.5561	**.5883**
DPBag-	100	50	.7407	.7901	.8018	.6527	.7684	.7932	.4325	.5492	.5701
		100	.7673	.7945	.8004	.7149	.7812	.7956	.4048	.5472	.5693
	150	50	.7717	.7952	.8159	**.7688**	.7852	.7963	.5036	.5367	.5803
		100	.7720	.7942	.8101	.7613	.7729	.7919	.4956	.5414	.5723
SAA	100	-	.7364	.7747	.7896	.6864	.7631	.7821	.3861	.5268	.5517
	150		.7693	.7879	.8014	.7540	.7717	.7889	.4971	.5466	.5748
NPB	1	-		.8713			.9245			.8199	

Table 8: Number of labels provided by each method using GBM before the privacy budget, ϵ, is used up on the UCI Adult Dataset with $\delta = 10^{-5}$. Note that DPBAG- and SAA have the same, data-independent privacy analysis and so provide the same number of labels as each other.

Models	n	k	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$
DPBag	100	50	51	413	1037
		100	52	413	1042
	150	50	115	931	2323
		100	117	933	2336
SAA	100	-	44	338	843
	150		95	759	1897

4 Pseudo-code for Subsample-and-aggregate

Algorithm 1 Semi-supervised differentially private knowledge transfer using subsample-and-aggregate

Input: ϵ, δ, $\mathcal{D}$, batch size n_{mb}, number of teachers n, noise size λ, maximum order of moments to be explored, L, unlabelled public data $\mathcal{D}_{pub}$
Initialize: $\{\theta_T^i\}_{i=1}^n$, θ_S, $\hat{\epsilon} = 0$, $\alpha(l) = 0$ for $l = 1, ..., L$
Partition the dataset n disjoint subsets $\mathcal{D}_i$, $i = 1, ..., n$ such that $\bigcup_i \mathcal{D}_i = \mathcal{D}$ and $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ for all i, j
while Teachers have not converged **do**
    for $i = 1, ..., n$ **do**
        Sample $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_{n_{mb}}, y_{n_{mb}}) \overset{\text{i.i.d.}}{\sim} D_i$
        Update teacher, T_i, using SGD
        $\nabla_{\theta_T^i} - \left[\sum_{s=1}^{n_m b} \sum_{c \in \mathcal{C}} y_{s,c} \log(T_{i,j}^c(\mathbf{x}_s))\right]$ (multi-task cross-entropy loss)
while $\hat{\epsilon} < \epsilon$ **do**
    Sample $\mathbf{x}_1, ..., \mathbf{x}_{n_{mb}} \sim \mathcal{D}_{pub}$
    for $s = 1, ..., n_{mb}$ **do**
        for $c \in \mathcal{C}$ **do**
            $n_c \leftarrow |\{(i, j) : T_{i,j}(\mathbf{x}_s) = c\}|$
        $r_s \leftarrow \arg\max\{n_c + Y_c : c \in \mathcal{C}\}$ where Y_c are i.i.d. $Lap(\frac{1}{\lambda})$
        Update the moments accountants
        for $l = 1, ..., L$ **do**
            $\alpha(l) \leftarrow \alpha(l) + 2\lambda^2 l(l+1)$
    Update the student, S, using SGD
    $\nabla_{\theta_S} - \sum_{s=1}^{n_{mb}} \sum_{c \in \mathcal{C}} r_{s,c} \log S^c(\mathbf{x}_s)$ (multi-task cross-entropy loss)
    $\hat{\epsilon} \leftarrow \min_l \frac{\alpha(l) + \log(\frac{1}{\delta})}{l}$
Output: S

References

[1] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

Chapter 6

Conclusion

In this thesis we introduced several novel GAN architectures to address a range of problems: treatment effect estimation, imputation, knockoff generation (for feature selection) and private synthetic data generation. In addition, we developed a generalisation of the well-established subsample-and-aggregate framework used for differential privacy. Though each of these problems presents themselves clearly in the medical domain, our methodologies are general and can be applied to a variety of domains. The architectures are basic in the sense that they use simple fully connected networks for the most part, but this only serves to highlight the fundamental ideas being proposed. We believe our methodologies would work with alternative architectures. For example, one could consider replacing most of the fully connected networks with recurrent neural networks to apply our methods to time-series data, or convolutional networks for image data, etc.

In Chapter 2, we introduced GANITE and SCIGAN. GANITE leverages the GAN framework to learn to generate the counterfactual outcomes for an observational intervention dataset with potentially multiple categorical interventions. Existing approaches to treatment effect estimation focused on the binary setting, with GANITE being one of the first papers to naturally generalise to multiple treatments. In SCIGAN, we extend the ideas in GANITE to the continuous intervention setting, once again considering multiple treatments. To model the "dosage curves" we consider them as sets and use a set-based neural architecture for our discriminators, of which there is a hierarchy. We also provide theoretical results in SCIGAN that apply not only to our SCIGAN framework but also to the original GANITE framework. Due to the nature of the problem, we demonstrate the efficacy of our methods on a mixture of real and semi-synthetic datasets.

In Chapter 3, we introduced GAIN. GAIN is a GAN-based imputation framework that learns to generate imputations for missing data by adversarially training

a discriminator to identify which components of an imputed vector are real (from the original data) and which are generated by our GAIN generator. Leveraging a hint mechanism to aid in convergence, GAIN outperforms state-of-the-art imputation methods such as MICE and through its generative nature can naturally perform multiple imputations, a particularly desirable property of imputation architectures, allowing uncertainty in the imputations to be captured by downstream models. We provide a theoretical analysis of GAIN, demonstrating that under similar conditions to the original GAN framework (e.g. sufficient capacity, sample size goes to infinity), we can recover the original data generating mechanism. Though the paper and theory are based in the missing completely at random (MCAR) setting, we empirically test it in the missing at random (MAR) and missing not at random (MNAR) settings.

In Chapter 4, we introduced KnockoffGAN. KnockoffGAN addresses a critical component of the FDR-controlled feature selection methodology *knockoffs*, by proposing an assumption-free knockoff generation model. In previous works, knockoff generation was limited to known distributions such as Gaussian or hidden Markov models. In KnockoffGAN we proposed a GAN framework capable of learning to generate appropriate knockoffs for any distribution that a GAN is capable of learning.

In Chapter 5, we switched focus to privacy, introducing PATEGAN and Differentially Private Bagging. In PATEGAN we approached the privacy problem by proposing to generate differentially private synthetic data. To do so, we leveraged the subsample-and-aggregate framework to replace the standard GAN discriminator with a differentially private classifier, which required knowledge transfer to a student model in order to enable back-propagation. In Differentially Private Bagging, we generalised the subsample-and-aggregate framework, analysing the effect of partitioning a given dataset multiple times (rather than once, as in standard subsample-and-aggregate). We demonstrated both empirically and theoretically that this afforded us tighter privacy guarantees and improved utility over a single partition.

6.1 Possible Extensions

We now highlight some directions that we feel are particularly promising future directions for this research. Perhaps the most promising direction is in the advent of new architectures, such as those that we leveraged in Chapter 2, introduced in [71]. In particular, GAIN might benefit greatly by modelling the generator as a function on sets rather than a function on vectors.

All the work presented in this thesis focused on the static setting, but many medical datasets (and many datasets in other domains) present themselves as time-series datasets. While extending the works may mostly be a case of replacing the fully connected GAN architectures with RNNs (or LSTMs or GRUs or transformers), there is likely more to it than that, with methods such as SCIGAN suffering from time-varying confounders that may need to be accounted for. In the privacy setting of PATEGAN, care will need to be taken to define precisely the level and type of privacy (should privacy be considered per time-step or as a stream?).

In the realm of knockoff generation (Chapter 4), there has been some progress, since the research in this thesis, that enables quantifying how close the knockoffs are to being true knockoffs, and more importantly proposes an adjustment that can be made to account for "inaccurate" knockoffs. Combining such work with KnockoffGAN would present a very promising and powerful knockoff generation procedure that is not only assumption free, but also aware of the limitations of the GAN methodology and/or architecture choices that mean the knockoffs may not quite be perfect.

Finally, recent papers such as [45] have demonstrated that PATEGAN can be improved upon by applying PATE to a higher-dimensional object than we do in PATEGAN. In [45], they apply aggregate vectors that are the same dimension as the input features, rather than the single-dimensional output we aggregate in PATEGAN. Exploration of this trade-off would be very valuable. It seems obvious, for example, that applying PATE to the very high-dimensional network weights would yield far too high a privacy cost, but whether the correct balance is at the dimensionality of the features is not completely clear. By considering intermediate layers of the discriminator, one could investigate this trade-off more directly.

Bibliography

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[2] Amjad Almahairi, Sai Rajeshwar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data. In *International Conference on Machine Learning*, pages 195–204. PMLR, 2018.

[3] Martin Arjovsky and Léon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. In *Inernational Conference on Learning Representations*, 2017.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[5] Rina Foygel Barber and Emmanuel J Candès. A Knockoff Filter for High-dimensional Selective Inference. *The Annals of Statistics*, 47(5):2504–2537, 2019.

[6] Richard Bellman. Dynamic Programming. *Science*, 153(3731):34–37, 1966.

[7] Ioana Bica, James Jordon, and Mihaela van der Schaar. Estimating the Effects of Continuous-valued Interventions using Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, 2020.

[8] Mikołaj Bińkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *International Conference on Learning Representations*, 2018.

[9] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[10] Zdravko I Botev, Joseph F Grotowski, Dirk P Kroese, et al. Kernel Density Estimation via Diffusion. *The Annals of Statistics*, 38(5):2916–2957, 2010.

[11] Lei Cai, Hongyang Gao, and Shuiwang Ji. Multi-Stage Variational Auto-Encoders for Coarse-to-Fine Image Generation. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 630–638. SIAM, 2019.

[12] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode Regularized Generative Adversarial Networks. In *International Conference on Learning Representations*, 2017.

[13] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[14] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating Multi-label Discrete Patient Records using Generative Adversarial Networks. In *Machine Learning for Healthcare Conference*, 2017.

[15] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

[16] Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. Towards Diverse and Natural Image Descriptions via a Conditional GAN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2970–2979, 2017.

[17] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan Salakhutdinov. Good Semi-supervised Learning That Requires a Bad GAN. *Advances in Neural Information Processing Systems*, 2017.

[18] Frank Dellaert. The Expectation Maximization Algorithm. Technical report, Georgia Institute of Technology, 2002.

[19] Jeff Donahue and Karen Simonyan. Large Scale Adversarial Representation Learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[20] Cynthia Dwork, Aaron Roth, et al. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[21] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training Generative Neural Networks via Maximum Mean Discrepancy Optimization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, 2015.

[22] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. *arXiv preprint arXiv:1706.02633*, 2017.

[23] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[24] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[25] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved Training of Wasserstein GANs. *Advances in Neural Information Processing Systems*, 30, 2017.

[26] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Advances in Neural Information Processing Systems*, 2017.

[27] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of Convex Analysis*. Springer Science & Business Media, 2004.

[28] Michael P Holmes, Alexander G Gray, and Charles Lee Isbell. Fast Nonparametric Conditional Density Estimation. *arXiv preprint arXiv:1206.5278*, 2012.

[29] Felix Horger, Tobias Würfl, Vincent Christlein, and Andreas Maier. Deep Learning for Sampling from Arbitrary Probability Distributions. In *International Workshop on Machine Learning for Medical Image Reconstruction*, pages 129–137. Springer, 2018.

[30] Huaibo Huang, Ran He, Zhenan Sun, Tieniu Tan, et al. IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis. *Advances in Neural Information Processing Systems*, 31, 2018.

[31] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-Image Translation with Conditional Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[32] Unnat Jain, Ziyu Zhang, and Alexander G Schwing. Creativity: Generating Diverse Questions using Variational Autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6485–6494, 2017.

[33] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. KnockoffGAN: Generating Knockoffs for Feature Selection using Generative Adversarial Networks. In *International Conference on Learning Representations*, 2018.

[34] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *International Conference on Learning Representations*, 2018.

[35] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations*, 2018.

[36] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[37] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. In *International Conference on Machine Learning*, pages 1857–1865. PMLR, 2017.

[38] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2014.

[39] Diederik P Kingma, Danilo J Rezende, Shakir Mohamed, and Max Welling. Semi-supervised Learning with Deep Generative Models. *Advances in Neural Information Processing Systems*, 2014.

[40] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[41] Abhishek Kumar, Prasanna Sattigeri, and P Thomas Fletcher. Semi-supervised Learning with GANs: Manifold Invariance with Improved Inference. In *Advances in Neural Information Processing Systems*, 2017.

[42] Bruno Lecouat, Chuan-Sheng Foo, Houssam Zenati, and Vijay R Chandrasekhar. Semi-supervised learning with GANs: Revisiting manifold regularization. *arXiv preprint arXiv:1805.08957*, 2018.

[43] Yujia Li, Kevin Swersky, and Rich Zemel. Generative Moment Matching Networks. In *International Conference on Machine Learning*, 2015.

[44] Don O Loftsgaarden, Charles P Quesenberry, et al. A Nonparametric Estimate of a Multivariate Density Function. *The Annals of Mathematical Statistics*, 36(3):1049–1051, 1965.

[45] Yunhui Long, Suxin Lin, Zhuolin Yang, Carl A Gunter, Han Liu, and Bo Li. Scalable Differentially Private Data Generation via Private Aggregation of Teacher Ensembles. 2019.

[46] Chao Ma, Sebastian Tschiatschek, Konstantina Palla, José Miguel Hernández-Lobato, Sebastian Nowozin, and Cheng Zhang. EDDI: Efficient Dynamic Discovery of High-Value Information with Partial VAE. In *International Conference on Machine Learning*, 2019.

[47] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks. In *International Conference on Machine Learning*, 2017.

[48] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled Generative Adversarial Networks. In *International Conference on Learning Representations*, 2017.

[49] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*, 2014.

[50] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[51] In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47(1):90–100, 2003.

[52] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[53] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems*, 2016.

[54] Augustus Odena. Semi-Supervised Learning with Generative Adversarial Networks. *arXiv preprint arXiv:1606.01583*, 2016.

[55] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional Image Synthesis with Auxiliary Classifier GANs. In *International Conference on Machine Learning*, 2017.

[56] Ingram Olkin and Clifford H Spiegelman. A Semiparametric Approach to Density Estimation. *Journal of the American Statistical Association*, 82(399):858–865, 1987.

[57] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *International Conference on Learning Representations*, 2017.

[58] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434*, 2015.

[59] Ladislav Rampášek, Daniel Hidru, Petr Smirnov, Benjamin Haibe-Kains, and Anna Goldenberg. Dr. VAE: improving drug response prediction via modeling of drug perturbation effects. *Bioinformatics*, 35(19):3743–3751, 2019.

[60] Tim Salimans, Ian J Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, 2016.

[61] Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards Generation of Small Graphs using Variational Autoencoders. In *International Conference on Artificial Neural Networks*, 2018.

[62] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles A Sutton. VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. In *Advances in Neural Information Processing Systems*, 2017.

[63] George R Terrell and David W Scott. Variable Kernel Density Estimation. *The Annals of Statistics*, pages 1236–1265, 1992.

[64] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

[65] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An Uncertain Future: Forecasting from Static Images Using Variational Autoencoders. In *European Conference on Computer Vision*, pages 835–851. Springer, 2016.

[66] Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the Quantitative Analysis of Decoder-Based Generative Models. In *International Conference on Learning Representations*, 2017.

[67] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially Private Generative Adversarial Network. *arXiv preprint arXiv:1802.06739*, 2018.

[68] Jinsung Yoon, Lydia N Drumright, and Mihaela Van Der Schaar. Anonymization Through Data Synthesis Using Generative Adversarial Networks (ADS-GAN). *IEEE journal of biomedical and health informatics*, 24(8):2378–2388, 2020.

[69] Jinsung Yoon, James Jordon, and Mihaela Schaar. GAIN: Missing Data Imputation using Generative Adversarial Nets. In *International Conference on Machine Learning*, 2018.

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

[71] Manzil Zaheer, Satwik Kottur, Siamak Ravanbhakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J Smola. Deep Sets. In *Advances in Neural Information Processing Systems*, 2017.

[72] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based Generative Adversarial Network. In *International Conference on Learning Representations*, 2017.

[73] Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. *arXiv preprint arXiv:1703.10960*, 2017.

[74] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.