Generative adversarial networks in marketing: Overcoming privacy issues with the generation of artificial data

Gilian R. Ponte[a]

Supervisor: Jaap E. Wieringa[a]

[a]Affiliated with the Department of Marketing, Faculty of Economics and Business, University of Groningen, PO Box 800, 9700 AV Groningen, The Netherlands.

Privacy is a fundamental human right. Over the years, we observe an increase in privacy concerns due to the growing amount of data and the development of methodologies that pressure the fundamental right to privacy. We define generative adversarial networks (GANs) to represent *any* data generating process and generate privacy-friendly artificial data for three data sets, each with different characteristics. Our methodology stands out in terms of the ability to generate and share individual-level privacy-friendly data without sacrificing the ability to derive meaningful insights. We show that GANs generate artificial data that are useful in marketing modeling cases, while achieving anonymization without data minimization. We display that we are able to derive the same real marketing insights from artificial data. Surprisingly, we find that estimations on artificial data are able to outperform the real data in terms of predictive accuracy. We contribute to the marketing literature by showing that academics and firms are able to generate privacy-friendly artificial data, increase the predictive performance of estimations, promote data sharing even under GDPR requirements and accelerate scientific progress in and outside the field of marketing.

Key words: privacy, data protection, generative adversarial networks and artificial intelligence

1. Introduction

Marketing has a rich history of modeling data in efforts to understand customer behaviour. Wedel and Kannan (2016) present an outline of the timeline of marketing data and analytics. As new types of data became available, the development of new methods naturally followed. Parallel to the emergence of new data types and methods, the volume of data increased. The rise and popularization of the internet and emergence of social media are a considerable game changer for firms and academics to collect data, and a resource to rich data sets containing detailed information on individual activities of users (Bucklin and Sismeiro 2009, Goodfellow et al. 2016). Until recently, methods from the artificial intelligence (AI) literature started to outperform humans on complex tasks. For example, deep learning models have reached at least human performance on an image recognition task with 1.2 million high-resolution images (Krizhevsky et al. 2012), playing video games such as Atari (Badia et al. 2020), defeating the world champion in the game of Go (Silver et al. 2016, 2017) and 48-hours early detection of acute kidney disease (Tomasev et al. 2019). As of only 2016, a rule of thumb is that supervised deep learning algorithms achieve at least human performance when trained with at least 10 million labelled observations (Goodfellow et al. 2016). As of now, marketing science has not reaped the full benefits from such algorithms to overcome marketing issues.

The marketing literature has defined privacy as one of its biggest priorities and the importance of privacy in marketing continued to increase over the years (Wedel and Kannan 2016, Rust 2019, Wieringa et al. 2019, Marketing Science Institute 2020). Annually, firms spend around 36 billion dollars to capture and leverage customer data (Columbus 2014). The vast amount of investment in leveraging customer data combined with the growth of data and possibilities to capture individual customer data lead to privacy concerns among individuals (Acquisti et al. 2015, 2016, Martin et al. 2017). Subsequently, privacy concerns under individuals may lead to a decrease in firm- and industry performance, effective personalization, willingness to disclose information and an increase in regulatory oversight (Goldfarb and Tucker 2011, van Doorn and Hoekstra 2013, Martin et al. 2017). In an attempt to protect the privacy of the individual, the European Commission created privacy legislation in the form of the General Data Protection Regulation (GDPR) (European Commission 2012). This implies that for academia and practice to conform to privacy legislation, we are limited as marketing science to derive meaningful insights from data.

To highlight the importance of our study, we provide a recent example that illustrates the consequences of privacy concerns. Facebook and Cambridge Analytica shared data of more than 87 million users without the consent of its customers (Reuters 2019). Data sharing enables firms to collaborate, enrich data sets to get a better customer view and enhance the predictive power of estimations (Peltier et al. 2013). The risks of such endeavours are clear. Firms are at risk for potential privacy costs, losses in brand value, legal fines or customer trust (Schneider et al. 2017, Martin et al. 2017). In case of Facebook and Cambridge Analytica, privacy concerns led to a decrease of 119 billion dollars' in Facebook's market value, a 5 billion dollar fine from the Federal Trade Commission (FTC) and the termination of Cambridge Analytica (Neate 2018). Unfortunately, the Facebook and Cambridge Analytica data breach is no exception. Wedel and Kannan (2016) describe that in the last decade 5,000 major data breaches occurred with an average cost of four million dollars. For example, the firms Zynga, Quora and Equifax disclosed a data breach of 218 million users, 100 million users and 145 million users, respectively (Mathews 2017, McLean 2018, Zynga 2019).

This study is one of the first marketing studies to reap the benefits from the developments in the AI literature. Our work is heavily inspired by the AI literature on generative modeling. Goodfellow et al. (2014a) introduce a theoretical proof for GANs to represent *any* data generating process. Goodfellow et al. (2014a) describe that we are able to sample directly from the data generating process to generate artificial samples. Theoretically, GANs are able to generate an infinite amount of data that retains the ability to derive meaningful insights (Goodfellow et al. 2014a). However, GANs are in a very early stage of development (Goodfellow 2016). In this study, we introduce GANs to investigate the ability to generate privacy-preserving marketing data. Subsequently, our research question is: *"How are GANs able to generate privacy-preserving marketing data that maintains the ability to derive meaningful insights?"*. We address the Marketing Science Institute (2020) research priority to fundamentally alter the raising concerns about data privacy. Specifically, we aim to resolve the trade-off between privacy and prediction and show that prediction can benefit from anonymization.

In this study, we empirically demonstrate how GANs are able to generate artificial data from three marketing data sets, each with different characteristics. First, we show that GANs are able to represent the real data distributions and relationships among variables. Secondly, we provide evidence for the predictive validity of the artificial data. Interestingly, artificial estimations occasionally outperform the real estimations in terms of predictive accuracy. Finally, we show that the parameter estimates of real estimations are largely maintained in the artificial estimations. Therefore, we show that artificial data are useful for academics to accelerate the development of theory by sharing data. For example, academics are able to publish the artificial data together with a Marketing Science publication to allow reproducibility of the study. Or in the case of practitioners, where the artificial data help to increase the predictive validity of existing models. To set an example, accompanied with our manuscript, we publish all artificial data, code, trained models and results.

Our paper makes several contributions to the literature of privacy in marketing. First, we provide theoretical and empirical arguments which lead to the development of GANs to successfully generate privacy-friendly artificial data, while maintaining the ability to derive meaningful insights from artificial data (Wedel and Kannan 2016, Rust 2019, Wieringa et al. 2019). Empirically, we show that artificial data increase the ability of practitioners to predict the behavior of individuals or sales of a firm, while preserving the privacy of individuals. Moreover, we find that parameter coefficients of estimations on artificial data closely resemble the real data parameter coefficients. Therefore, other academics are able to reproduce results and improve the generalizability of a study's findings. As marketing science produces more generalizable results, the development of theory or scientific progress accelerates. Alternatively, more general empirical tests can investigate the validity of existing theories.

Secondly, the marketing literature on the generation of privacy-friendly data largely relies on methods where the quality of the samples decrease as the dimensionality of the data increases. Moreover, we are unable to reliably monitor when the samples are of high-quality and often have to make assumptions a priori about the relationships between variables in the data (Danaher and Smith 2011, Holtrop et al. 2017, Schneider et al. 2017, 2018). This study provides empirical evidence for the performance of a variety of GAN architectures to generate marketing data. We show that we are able to effectively monitor the performance of GANs in terms of speed of convergence and the ability to generate high-quality samples. We demonstrate that GANs are able to scale to the generation of high-dimensional data and can approximate complex multivariate distributions very well.

Finally, we also contribute to literature outside the field of marketing. At the time of writing, the AI literature is heavily focused on the generation of high-dimensional data (e.g., images), while marketing data consist of rather low-dimensional consumer data (Salimans et al. 2016, Gulrajani et al. 2017, Karras et al. 2017). Therefore, architectural choices are likely to yield different results. We perform robustness checks to identify the hyperparameters that are available in GANs to increase the quality of the artificial data and reflect on our findings with recent AI literature. Furthermore, Acquisti et al. (2015) take a broader perspective on the consequences of privacy concerns. Privacy concerns and the inability to share data prevent early identification of infectious disease outbreaks, progress in genetic research (e.g., cancer or Parkinsons disease), access to credit, identity theft and the unintended creation of monopolies (e.g., Facebook and Google). Therefore, the results of this study possibly contribute to progress in other scientific fields.

The rest of this study is presented as follows. In the subsequent chapter, we highlight the privacy-preserving literature in marketing. We describe previous attempts to preserve the privacy of individuals, while maintaining the ability to derive insights. In chapter 3, we give an overview of several state-of-the-art architectures of GANs. We describe two neural networks that function as two competing players in a GAN. Moreover, we describe the developments that lead to the stable convergence of a GAN. In chapter 4, we describe the advantages of deep neural networks and provide a theoretical analysis of the approximation of the artificial data distribution to the real data distribution. In chapter 5, we describe our empirical analysis for the three data sets, compare GAN architectures and provide robustness checks for the hyperparameters present in GANs. In chapter 6, we describe the results of our empirical analysis. Finally, we conclude and provide future research directions in chapter 7.

2. Related work

To cope with privacy issues and maintain the ability to identify individuals, Wedel and Kannan (2016), European Commission (2012) identify two potential actions to take: data minimization and data anonymization. Data minimization is to limit the collection of data and the disposal of redundant data. This may impede the goal of academics to generate generalizable results. Data anonymization can be accomplished with non-model-based approaches such as k-anonymization, removing personal identifiable information, recoding, swapping, or randomizing data or hashing algorithms (Reiter 2010, Wieringa et al. 2019). In this literature review, we focus on model-based approaches to data protection. We argue that non-model-based methods often destroy the ability to derive insights from the data and the artificial data remains privacy sensitive. Whereas, model-based approaches allow theoretical privacy guarantees (Miller and Tucker 2011, Dwork and Roth 2014, Schneider et al. 2017). We describe several attempts for model-based data anonymization to preserve the privacy of individuals in the marketing literature (Danaher and Smith 2011, Holtrop et al. 2017, Schneider et al. 2017, 2018).

Early in the literature, Danaher and Smith (2011) introduce copula models to successfully generate high-dimensional data and account for the complex mixture of continuous and discrete variables in marketing data distributions. Previous methods were less able to account for such complex distributions. For example, previous methods rely largely on the marginal distributions to have fixed distributions. Compared to GANs, we are not able to reliably estimate copula models with maximum likelihood estimation. Instead, we are required to use Markov Chain Monte Carlo (MCMC) sampling to generate samples of high-quality. Unfortunately, this introduces difficulties with respect to the convergence of Markov chains, which GANs are designed to overcome (Goodfellow 2016). Moreover, the paper of Danaher and Smith (2011) does not focus on the generation of privacy-preserving data.

Holtrop et al. (2017) develop a generalized mixture of Kalman filters (GMOK) model to capture customer clusters and the dynamic relationships in data over time. As a result, it is not necessary to store data of individual customers. The main disadvantage of their method is that the training data contains real identifiable individuals and we are only able to store the parameter estimates over time. Therefore, individual-level data sharing is impossible and the predictive accuracy of the estimation decreases over time. Although the estimation outperforms existing methods in terms of the predictive accuracy staying power.

Schneider et al. (2017) segment data from a ticketing company with around 95,000 customers based on annual spend at the company. The authors draw samples from a Dirichlet-multinomial model with a privacy parameter k to generate privacy-friendly data for each segment. A high k results in a small level of privacy protection and a low k results in a high level of privacy protection. Unfortunately, when the authors try to identify the segment of artificial customers, a low parameter k substantially reduces the ability to derive meaningful insights.

Schneider et al. (2018) develop a model on AC Nielsen point of sales data. The authors rely on a Bayesian specification of the SCAN*PRO model with a privacy protection parameter k. The authors obtain the posterior predictive distribution of the retail sales which contains protected model parameters and sample from this protected distribution with MCMC sampling. The authors describe the inclusion of only a select number of variables as the main limitation. We argue that the parameter estimates that result in the posterior predictive distribution are not robust to violations of the Gauss Markov theorem. For example, endogeneity issues such as omitted variable bias are a reason for concern and the variance of the parameter estimates might be biased in case of the potential violation of normality, heteroskedasticity or autocorrelation. As a result, to sample from the posterior predictive distribution might lead to inaccurate artificial data. Finally, the authors are required to specify the relationships of the variables on the dependent variable a priori. Therefore, the authors are required to make assumptions about the relationships of the variables.

We provide a summary of the relevant literature in Table 1. Here, *Method* refers to the method that is used to generate data, *Multivariate* refers to the fact whether we are able to account for the relationships between the variables in the artificial data. *Case* shows for which marketing case the artificial data can be used. *Assumptions* refers to whether we are required to make assumptions about the variables prior to data generation. *Data sharing* refers to the fact whether we are able to share the artificial data with others.

Table 1 An overview of the model-based (privacy-preserving) marketing literature.

Study	Method	Multivariate	Case	Assumptions	Data sharing	Privacy-preservation
Danaher and Smith (2011)	Copula models	✓	Any	✓	✗	✗
Holtrop et al. (2017)	GMOK	✓	Churn prediction	✓	✗	Store parameter estimates over time.
Schneider et al. (2017)	MCMC	✗	Clustering	✓	✓	Protect customer segment index.
Schneider et al. (2018)	MCMC	✗	SCAN*PRO	✓	✓	Sample protected store sales.
This study	GANs	✓	Any	✗	✓	Generate artificial data that do not exist.

The methodology that we propose differs from previous marketing privacy research as follows. First of all, GANs are known to generate high-quality samples that do not exist in reality (Radford et al. 2015, Karras et al. 2017). To illustrate, a GAN maps a noisy sample from a standard normal distribution to an artificial sample. Once the GAN learns weights to map the noise to high-quality samples, we are able to explore the latent space of the GAN defined by its weights. In case of GANs, Goodfellow et al. (2016) describe that we can refer to this latent space as a manifold (see section 4.2). An important characteristic of the manifold is the set of its tangent planes. The set of tangent planes of the manifold span the local directions of variation allowed by the manifold. These local directions tell us how we are able to move on the manifold, while maintaining the ability to generate high-quality samples. Radford et al. (2015) make use of this property and show that we are able to "walk" into the directions of variation allow by the manifold to arrive at images that do not exist. The authors show that we are able to do vector space arithmetic with faces of individuals. To illustrate, the authors subtract the image of a male without glasses from a male with glasses and add an image of a female to arrive at an image of a female wearing glasses which was not available in the training data set.

Second, a GAN consists of a competition between two neural networks. Neural networks are universal function approximators (Leshno et al. 1993). The universal approximation theorem implies that a neural network is able to resemble any data generating process to any degree of accuracy of a specific data set. Therefore, we directly generate data from the data generating process instead of a prespecified specification. As a result, our artificial data are robust to potential violations of econometric assumptions or specification errors. In contrast to previous research, we are not required to specify the relationship among the variables from the data set a priori (see Table 1). For example, Schneider et al. (2018) only describe data generation for a SCAN*PRO model. Therefore, it is not possible to share the data with academics that might have other goals in mind. On the other hand, GANs are theoretically able to fully restore the relationships in a high-dimensional multivariate distribution of variables to an arbitrary *non*-linear degree. As a result, we are able to share the artificial data from a GAN for any goal the practitioner or academic has in mind.

Finally, the marketing literature largely relies on Markov chain approximation to generate artificial data (Danaher and Smith 2011, Schneider et al. 2017, 2018). Goodfellow (2016) describe that Markov chains do not guarantee to be a representative sample from the target distribution. There is no way to test whether a Markov chain has converged and as a result the samples are often used to early to be a representative sample from the target distribution. Theoretically, we only know that the Markov chain will converge to equilibrium distribution, just not when (Goodfellow et al. 2016). This paper proposes GANs that consist of training two neural networks using maximum likelihood estimation (Goodfellow 2016). Compared to Markov chains, maximum likelihood estimation allows us to track the progress the two neural networks make during training (see section 6.4.2). Moreover, GANs are able to take advantage of the universal function approximation theorem to scale to much higher dimensions (Goodfellow 2016). For example, the generation of high-quality pixel images of celebrity faces which are in $\mathbb{R}^{1024\times1024\times3}$ (Karras et al. 2017). In the next chapter, we describe how GANs deal with high-dimensional input spaces.

3. Generative adversarial networks

A Generative Adversarial Network (GAN) is described by the idea of a game between two players (Goodfellow 2016). The first player is a generator. Let Z be a standard normal random variable with the distribution $p_Z(\boldsymbol{z})$. The generator takes a random noise sample z from the distribution p_Z. The generator has the goal to approximate the real data distribution or ground truth as close as possible. The second player is a discriminator, that distinguishes samples to be from the generator or real data distribution. This game is illustrated by a competition between a counterfeiter, the generator, that tries to create a real image, and the discriminator acts as the art critic to identify the counterfeit images (see Figure 1). The goal of the game is for the generator to create artificial samples of sufficient quality that fool the discriminator. When the generator succeeds, the images from the generator resemble the real images. The strength of the discriminator to separate samples affects the ability of the generator to generate high-quality samples. Similarly, the strength of the generator to generate realistic samples affects the strength of the discriminator. It is this competition that drives the players to improve both respective performances.

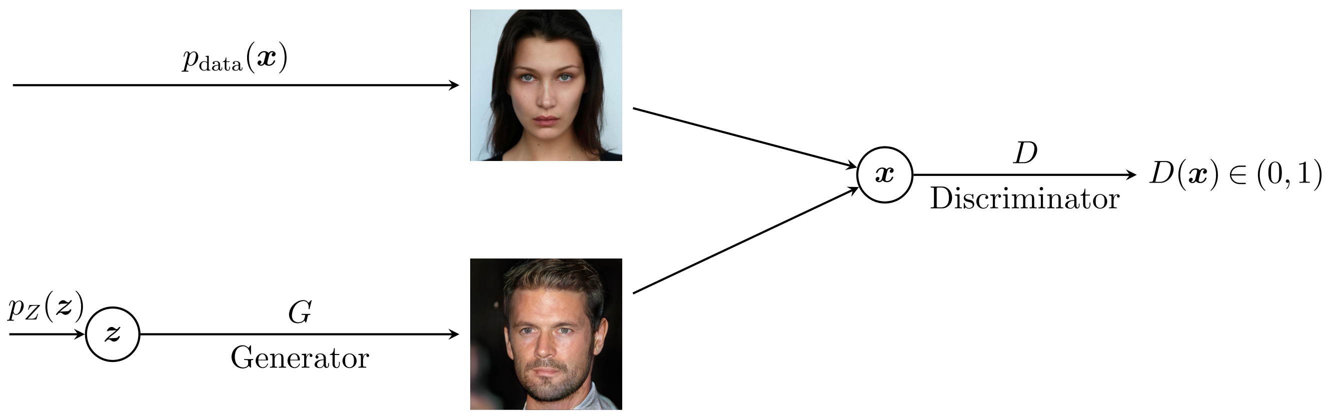

Figure 1 **A simple visualization of the architecture of a GAN. The generator creates artificial samples that are combined with real data samples. This input is used for the discriminator, which decides whether the sample is from the real or generator's distribution. Image of artificial face taken from Karras et al. (2017).**

3.1. Formal objective

Formally, both players are functions represented by neural networks (Goodfellow et al. 2014a). Let a neural network N admit n input units and k output units that is parameterized by its weights $\boldsymbol{\theta}$. Thus a neural network is a function $N_\theta : \mathbb{R}^n \rightarrow \mathbb{R}^k$. The generator is defined as a neural network G that maps noise samples $\boldsymbol{z}$ into artificial samples $G(\boldsymbol{z})$ to fool the discriminator. Formally, we can think of G as a random variable that is a function $G : \mathbb{R}^n \rightarrow \mathbb{R}^k$. The distribution of G is defined as p_G.

Let p_{data} denote the distribution of the data generating process, real distribution or ground truth. The discriminator is defined as a neural network D that takes samples of equal size from p_{data} labelled with 1 and p_G labelled with 0 from the generator as inputs. Similarly, we can think of the discriminator as a random variable D that is a function $D : \mathbb{R}^n \rightarrow [0,1]^k$. The discriminator predicts the probability of a sample to be from p_{data} or from p_Z. The distribution of D is defined as p_D. These functions compete in a minimax game (Goodfellow et al. 2014a):

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log(D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}))]. \tag{1}$$

In Equation 1, D has the objective to maximize the probability that the real samples are real, while G has the objective to minimize the probability of the real samples evaluated by D to be real. In other words, the objective of G is to minimize the maximum attainable of D. In case of convergence, the minimax game results in p_G to be equal to the data generating process p_{data}. As a result, the generator generates realistic artificial samples (i.e., $p_G = p_{\text{data}}$).

In the minimax game, the value function $V(D,G)$ can be interpreted as three-dimensional space with a loss surface that depends on the weights of the discriminator $\boldsymbol{\theta}^{(D)}$ and the weights of the generator $\boldsymbol{\theta}^{(G)}$. On this surface, the generator minimizes the maximum attainable of the discrimination for the value function. $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}$ refers to expected samples from the probability distribution of the data generating process and $\mathbb{E}_{\boldsymbol{z}\sim p_Z(\boldsymbol{z})}$ refers to expected samples from a standard Gaussian (i.e., $\boldsymbol{z} \sim N(0,1)$). $G(\boldsymbol{z})$ refers to the output of the generator with noise samples $\boldsymbol{z}$. $D(G(\boldsymbol{z}))$ refers to the probabilities that the artificial samples are from the real distribution. $D(\boldsymbol{x})$ refers to the probability estimates of the discriminator that samples are from the real distribution.

To learn the generator's distribution p_G, G takes samples of noise $\boldsymbol{z}$ with parameters $\boldsymbol{\theta}^{(G)}$. Generally, the generator outputs a sample $G(z)$ from the distribution p_G in a range of a tanh activation function. In other words, in a range of $(-1, 1)$. Although this restricts the distribution of the artificial data distribution, we align the scaling of the real data distribution with feature scaling. Before we feed the real data to the GAN, we scale the real data distribution in a range of $(-1, 1)$. This does not only align the distribution of p_{data} with that of p_G, it also results in faster convergence of the two networks (LeCun et al. 1998). To learn the discriminator's distribution p_D, D takes a combination of the real and artificial samples $\boldsymbol{x}$ with parameters $\boldsymbol{\theta}^{(D)}$ and outputs a probability between 0 and 1 (see Figure 1).

3.2. Loss functions

Both players in the minimax game employ maximum likelihood estimation. For example, we can see maximum likelihood as an attempt to let p_D approximate p_{data}. In this study, we refer to this distance between p_{data} and p_D as the negative log-likelihood. In the literature, minimizing the negative log-likelihood is equivalent to minimizing the Kullback-Leibler divergence or cross-entropy and are used interchangeably (Bishop 2006). In the next paragraphs, we describe the loss functions of the discriminator and generator in detail.

3.2.1. Discriminator. Consider the discriminator D, the neural network aims to determine whether a sample is from p_{data} or p_G (Goodfellow et al. 2014a). From Equation 1, we observe that the discriminator has the objective to maximize the log-likelihood of observing a sample from p_{data}. Simultaneously, the discriminator has the objective to maximize the log-likelihood of *not* observing a sample from p_G.

The discriminator uses a sigmoid activation function to map the input to a probability to determine whether the sample is from p_{data} or p_G in a range of $(0, 1)$. Compared to the mean squared error, taking the log-likelihood has the attractive property that it removes the exp(.) from the sigmoid activation function $\frac{1}{1+e^{-x}}$. Thereby, we prevent the loss function from saturation (Bengio et al. 1994). Therefore, the loss function of the discriminator is defined as follows:

$$J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -\frac{1}{m}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log(D(\boldsymbol{x})] - \frac{1}{m}\mathbb{E}_{\boldsymbol{z}\sim p_Z}[\log(1 - D(G(\boldsymbol{z}))]. \tag{2}$$

In Equation 2, let m be the number of observations in a mini-batch. The mini-batches consist of a random uniform sample of m observations from p_{data} and p_Z. Or in other words, the mini-batch contains a random sample of a combination of observations from p_G and p_{data}. We introduce mini-batches for three reasons. First of all, random mini-batches lead to a noisy estimate of the expected gradient of the loss function with respect to the weights, which is helpful to escape local minima, saddle points or plateaux in the optimization procedure, because each time we draw a mini-batch the loss function is different (Dauphin et al. 2014). As a side note, in this paper we refer to the gradient of the loss function with respect to the weights simply as the gradient. Smaller mini-batches in terms of the number of observations are described by Wilson and Martinez (2003) to have a regularizing effect, which prevents overfitting (also see section 6.5). Finally, the computational cost to obtain the gradient is linear in the number of observations (Goodfellow et al. 2016). Therefore, in the case of substantially large data sets, it is computationally too expensive to use all observations for a weight update. Additionally, Goodfellow et al. (2016) describe that from the standard error of the mean gradient from n observations is defined as $\sigma/\sqrt{n}$. As a result, there are less than linear returns on increasing the number of observations n mini-batch (i.e., diminishing returns). Therefore, the mini-batch size is a trade-off between an accurate estimate of the gradient and computational efficiency. Usually, 50 to 256 observations are used to obtain an estimate for the expected gradient (Goodfellow et al. 2016).

3.2.2. Generator. From Equation 1, we derive that the generator G attempts to minimize the maximum attainable of the discriminator D. In other words, G seeks to make the discriminator D believe that the artificial samples $G(\boldsymbol{z})$ from p_G are real. Note that the generator does not make reference to p_{data} directly, but only to p_Z (see Equation 1). Goodfellow (2016) argues that this makes the generator resistant to overfitting. To illustrate, if the generator would take samples from p_{data} directly, the generator could just learn an identity function to generate realistic samples. In the case where we make indirect reference to p_{data}, we force the generator to learn the same probability distribution, but not necessarily the exact observations.

In practice, to minimize the objective of the generator from Equation 1 is not ideal in training (Goodfellow 2016). In an initial stage of training, the discriminator is able to separate between a sample from p_{data} and p_G with very high confidence, because p_G still very much resembles the noise from p_Z. This results in the term from Equation 1, $\mathbb{E}_{\boldsymbol{z}\sim p_Z}[\log(1 - D(G(\boldsymbol{z}))]$ to be zero or very close to zero. As a result, the generator does not have a loss function to derive the gradient. This results in vanishing gradients for the generator.

Goodfellow (2016) reformulates the loss function of the generator G to maximize $\log(D(G(\boldsymbol{z})))$ instead of minimizing $\log(1–D(G(\boldsymbol{z}))$. After reformulation, the generator has the goal to increase the log probability that the discriminator makes a mistake, instead of decreasing the log probability that the discriminator makes the correct prediction (Goodfellow 2016). This enables the generator to have a more stable gradient throughout the minimax game (Goodfellow et al. 2016). To illustrate, consider an initial state of training where the discriminator is highly confident that $G(\boldsymbol{z})$ are artificial (i.e. $D(G(\boldsymbol{z}))=0$). The term $\mathbb{E}_{\boldsymbol{z}\sim p_Z}[\log(D(G(\boldsymbol{z}))]$ is now closer to 1 rather than 0. Therefore, to maximize $\log(D(G(\boldsymbol{z})))$ makes it unlikely that loss function for the generator vanishes. This leads to the following negative log-likelihood of the generator, where we apply the log function to prevent the loss function from saturating:

$$J^{(G)}(\boldsymbol{\theta}^{(D)},\boldsymbol{\theta}^{(G)})=-\frac{1}{m}\mathbb{E}_{\boldsymbol{z}\sim p_Z}[\log(D(G(\boldsymbol{z}))]. \tag{3}$$

Here, G has the objective to minimize the negative log-likelihood of D over $G(\boldsymbol{z})$ over m observations in a mini-batch. Both loss functions $J^{(G)}$ and $J^{(D)}$ are heavily adopted throughout the literature of GANs (e.g. Radford et al. 2015, Karras et al. 2017, Kumar et al. 2018, Beaulieu-Jones et al. 2019).

3.3. Convergence

Goodfellow et al. (2016) describe that the size and non-linearity of neural networks result in a high-dimensional non-convex loss function for both players. Specifically, if we have k-dimensional parameter vectors, we have a k-dimensional loss surface in $\mathbb{R}^{k+1}$. To illustrate, the language model GPT-3 from Brown et al. (2020) contains 175 billion parameters trained on a data set with nearly a trillion words. Therefore, the graph of this loss function is in $\mathbb{R}^{1.75e+11}$.

Generally, the structure of the graph of a loss function in such a high-dimensional space is poorly understood (Choromanska et al. 2014). What we do know is that the high-dimensional space of the loss function is rich in local minima, plateaux and saddle points. Therefore, we are not guaranteed to converge to a global minimum (Choromanska et al. 2014, Dauphin et al. 2014). Interestingly, Choromanska et al. (2014) show theoretically and empirically that for a sufficiently large neural network it is not crucial to find a global minimum and a local minimum or not even such a critical point suffices. Specifically, Choromanska et al. (2014) show that as the number of hidden units increases, the mean and variance of the loss function values decrease. The authors argue that to recover a global minimum often leads to overfitting. To illustrate the importance of overfitting, consider the discriminator from a GAN. If we would only settle for a global minimum of the discriminator, the discriminator would predict every sample correctly and leave no gradient for the generator to improve the artificial samples. Compared to shallow neural networks where the variance of the values for the loss function is larger, the global minimum that is so heavily desired is often very closely surrounded by acceptable local minima in deep networks. This indicates that recovering a global minimum is more important for shallow networks compared to deep networks.

Goodfellow et al. (2016) describe that optimization algorithms to train neural networks are based on stochastic gradient descent. To optimize the loss functions from Equations 2 and 3, Goodfellow (2016) recommends the usage of the stochastic gradient-based adaptive optimization algorithm Adam (Kingma and Ba 2014). With Adam, the concept of an adaptive learning rate is applied (see Algorithm 1). Adam scales the learning rate inverse proportional to the sum of squared gradients. Therefore, large partial derivatives of the loss function with respect to the weights receive the greatest decrease in learning rate. Unfortunately, the accumulation of large gradients leads to an early decrease in the learning rate. Therefore, Kingma and Ba (2014) define the forgetting factors β_1 and β_2 to exponentially decay the first and second moment estimates of the gradient. In this way, more recent gradients have more influence on the adjustment of the learning rate. In Algorithm 1, the gradient g_k is exponentially decayed by the hyperparameters β_1 and β_2. Together with these forgetting factors, the Adam algorithm defines a first and second moment estimate of the gradient m and v at iteration k. We use the moments to update the weights with the adjusted learning rates. Kingma and Ba (2014) observe that the moments are biased in early stages of training, because the two moment vectors are initialized as zero's. Therefore, the authors propose a bias correction. We summarize the training procedure of a GAN with Adam in Algorithm 1.

Algorithm 1 Training of the discriminator and generator with Adam optimizer. In our experiments we used $k = 1$, $\alpha = 0.001$, $\beta_1 = 0.5$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ to prevent division by zero (Kingma and Ba 2014). Here, $\odot$ is an element-wise operation.

```
Require: initialize $\boldsymbol{\theta}^{(G)}$, $\boldsymbol{\theta}^{(D)}$ (Glorot and Bengio 2010).
Require: $N$ training iterations or epochs, $k$ steps.
Require: $\alpha \in [0,1)$: learning rate
Require: $\beta_1$, $\beta_2 \in [0,1)$: exponential decay rates for moment estimates
Require: $m_0 \leftarrow 0$ (Initialize 1st moment vector)
Require: $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
1: for number of training iterations do
2:     for $k$ steps do
3:         Sample mini-batch of $m$ samples $z_i$ from $p_G$.
4:         Sample mini-batch of $m$ samples $x_i$ from $p_{\text{data}}$.
5:         Update the discriminator with Adam (Kingma and Ba 2014):
           $g_k \leftarrow \frac{1}{m}\sum_{i=1}^{m} \nabla_{\theta^{(D)}} \log D(x_i) + \log(D(G(z_i))$ (Obtain gradients)
           $m_k \leftarrow \beta_1 \cdot m_{k-1} + (1-\beta_1) \cdot g_k$ (Update biased first moment estimate (mean))
           $v_k \leftarrow \beta_2 \cdot v_{k-1} + (1-\beta_2) \cdot g_k \odot g_k$ (Update biased second moment estimate (variance))
           $\widehat{m}_k \leftarrow m_k/(1-\beta_1^k)$ (Compute bias-corrected first moment estimate)
           $\widehat{v}_k \leftarrow v_k/(1-\beta_2^k)$ (Compute bias-corrected second raw moment estimate)
           $\theta_k^{(D)} \leftarrow \theta_{k-1}^{(D)} - \alpha \cdot \widehat{m}_k/(\sqrt{v_k} + \epsilon)$(Update parameters element wise)
6:     end for
7:     Sample mini-batch of $m$ noise samples $z_i$ from $p_G$.
8:     Update the generator with Adam:
       $g_k \leftarrow \frac{1}{m}\sum_{i=1}^{m} \nabla_{\theta^{(G)}} \log(D(G(z_{(i)}))$
       ...
       $\theta_k^{(G)} \leftarrow \theta_{k-1}^{(G)} - \alpha \cdot \widehat{m}_k/(\sqrt{v_k} + \epsilon)$(Update parameters)
9: end for
```

3.4. Non-convergence

Theoretically, if both models have sufficient capacity, the competition between the two networks converges when equilibrium is accomplished (Goodfellow et al. 2014a). In a state of convergence, the generator produces highly realistic samples that make the discriminator unable to separate between a real sample from p_{data} and a sample from p_G. Unfortunately, Goodfellow et al. (2014a) describe that convergence is not guaranteed. Normally the minimax games establish equilibrium at minima for both the discriminator and generator. However, the discriminator and generator reduce their cost at the expense of each other (Goodfellow 2016). In other words, a decrease in the discriminator loss can result in an increase in the generator loss and vice versa. We demonstrate the oscillations in the loss function values visually in our robustness checks (see section 6.5). During training, the two players are very likely to settle at a local minimum or not even arrive at such a critical point. This highlights the importance of a delicate balance between the two players. To reach the state of equilibrium for both players was found to be very difficult and is subject to ongoing research (Goodfellow et al. 2016, Salimans et al. 2016, Arjovsky et al. 2017).

One of the "failure modes" of a GAN is mode collapse. The literature has identified mode collapse as the most important problem in the development of GANs (Goodfellow 2016). Where the quality of the output depends on how the minimax game develops. In case of mode collapse, the generator is only able to produce a subset of the real data distribution. Or in even more severe cases, only a single observation (Salimans et al. 2016). An explanation of why mode collapse occurs is the lack of diversity in the mini-batches that are provided to the discriminator, while in reality, the real distribution has a higher level of diversity (Goodfellow 2016). During a specific iteration, the discriminator trains on a low-in-diversity mini-batch. This makes the discriminator highly confident that only such observations exist. Therefore, the generator only has to generate a limited number of samples to fool the discriminator. In the next iteration of training, the discriminator has a different low diversity mini-batch and the generator adjusts the weights accordingly. This prevents the minimax game from converging (Salimans et al. 2016, Goodfellow 2016).

Furthermore, Theis et al. (2015) attempt to define a measure to evaluate the performance of generative models. The authors describe that the likelihood can be a misleading measure for sample quality. High likelihood does not imply that the samples from the generator are of high-quality, as low likelihood does not imply that the samples are of low quality (Theis et al. 2015, Goodfellow 2016). To illustrate, Theis et al. (2015), van den Oord and Dambre (2015) define an optimal (i.e., arbitrarily high likelihood) multivariate density $p(\boldsymbol{x})$ of random variable X and a pure noise distribution $q(\boldsymbol{x})$ (i.e., extremely low likelihood). Subsequently, the authors define a pdf $m(\boldsymbol{x})$ that is a mixture of $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$. Where we sample 99 percent of the times from $q(\boldsymbol{x})$ and 1 percent of the time from $p(\boldsymbol{x})$. In other words, a mixture model defined as: $.01p(\boldsymbol{x}) + .99q(\boldsymbol{x})$. The authors show that if we take the log of the likelihood of this mixture model, that the total log-likelihood only changes by a very small amount:

$$\log[0.01p(\boldsymbol{x}) + 0.99q(\boldsymbol{x})] = \log[p(\boldsymbol{x})] - 4.60517 + \log[q(\boldsymbol{x})] - .0100503 \quad (4)$$

This implies that in a worst-case scenario, the arbitrarily high log-likelihood decreases only with -4.60517. To illustrate the significance, the authors describe that the log-likelihood in case of image generation problems can become in the orders of 1000's (van den Oord and Dambre 2015). Furthermore, Theis et al. (2015) describe that the log-likelihood increases as the dimensionality of the data increases. Therefore, this issue becomes more prevalent for higher-dimensional data sets. As a result, a log-likelihood measure is discouraged for the quality of the samples from generative models. Theis et al. (2015) conclude that currently there is no state-of-the-art method or measure to evaluate the performance of a generative model during training and it depends on the application of the generative model.

3.5. Developments towards a stable GAN

In this section, we describe the attempts to allow a stable convergence of the minimax game. Moreover, we describe the attempts to improve the quality of the samples from a GAN. First, we describe hyperparameters such as label smoothing, batch normalization and dropout that help the minimax game in the GAN to converge in section 3.5.1. In section 3.5.2 and 3.5.3, we describe a modification of the GAN its loss function that promises a more stable convergence. In section 3.5.4, we describe convolutional networks that use a convolution operation to detect features in the data and its advantages. Finally, we describe networks that are able to deal with time-series data in section 3.5.5.

3.5.1. Hyperparameters. In general, the discriminator tends to minimize the loss function rapidly in an early stage of training, because the samples from the generator still very much resemble the noise. The result is that the gradient of the generator becomes zero or close to zero (i.e., $J^{(G)} \approx 0$). Similarly, mode collapse can also be a result of a highly confident discriminator. Label smoothing enables the discriminator to be less confident (Szegedy et al. 2016). To accomplish this, we transform the labels of the samples from

p_{data} and p_G in the mini-batches. For example, instead of a 0 and 1 indicating whether the samples are artificial or real, the labels are replaced by .1 or .9 (Szegedy et al. 2016, Goodfellow 2016). Therefore, the discriminator is less accurate and the generator is able to take a gradient step.

To increase the speed of convergence of neural networks, LeCun et al. (1998) propose to normalize the input data to a distribution that has a mean of zero (i.e., standard normally distributed). LeCun et al. (1998) illustrate the advantage of the normalization procedure with an example. Consider the situation where we feed all positive input values to the neural network, all the updates to the weights will be of the same sign. As a result, the optimization of the weights becomes inefficient and slow (LeCun et al. 1998). However, after many non-linear transformations (e.g., ReLU activation function) the activations of subsequent layers do not remain standard normally distributed.

Ioffe and Szegedy (2015) introduce batch normalization to increase the speed of learning in a neural network. To describe batch normalization, we first need to define activations of a neural network. Let the real-valued activation vector $\mathbf{x}$ at layer m of a neural network to be defined as follows: $\mathbf{x}^m = \sigma\left(\mathbf{W}^m\mathbf{x}^{m-1} + \mathbf{b}^m\right)$, where $\boldsymbol{b}^m$ is a bias vector at layer m, the mapping from the previous activation vector $\boldsymbol{x}^{m-1}$ to $\boldsymbol{x}^m$ is the weight matrix $\boldsymbol{W}$ and $\sigma()$ is a non-linear activation function applied element-wise (e.g., sigmoid) (Goodfellow et al. 2016). In a neural network, the activation vector a layer is also referred to as a representation.

The gradient of the loss function with respect to the weights tells us how to update each weight under the assumption that other weights do not change (Goodfellow et al. 2016). In actual fact, this assumption does not hold and the weights in other layers are adjusted simultaneously. Batch normalization scales the activations of the units *in each layer* to be from a standard normal distribution. In other words, the activations are centered around zero even after non-linear transformations in each layer. Therefore, batch normalization allows the network to learn at the same speed in each of its layers. Empirically, this allows the network to employ a higher learning rate, convergence faster and reduces the need for dropout (Ioffe and Szegedy 2015). We deal with dropout in the next paragraph. Additionally, Ioffe and Szegedy (2015) argue that batch normalization acts as regularization to prevent overfitting, because the transformation of the activations sometimes introduces noise in the learned representations. In the context of GANs, Radford et al. (2015) showed that the batch normalization prevents the generator from showing symptoms of mode collapse.

Finally, Srivastava et al. (2014) propose the key idea of dropout as input or hidden activations that are randomly multiplied with zero in the network for each time we sample a mini-batch. The idea of dropout is similar to bagging. When one neuron drops from the architecture a completely different network arises. However, bagging is only able to represent a linear number of networks and bagging is trained until convergence. Neural networks with dropout are able to represent an exponentially large number of networks, because the neural networks are distributed representations (see section 4.2). Compared to dropout, bagging is computationally more expensive, because we have to train each algorithm, such as a decision tree, until convergence. Finally, dropout causes the representations in the hidden layers to be a good representation in every subset of the network. Therefore, dropout acts as a regularization mechanism to increase the generalizability of the network, because some of the representations are deleted due to the multiplication of random activations with zero. Srivastava et al. (2014) show that dropout improves the generalization of neural networks dramatically. Nowadays, GANs benefit from dropout to decrease the certainty of the discriminator throughout empirical research (Beaulieu-Jones et al. 2019, Kumar et al. 2018).

3.5.2. Wasserstein GAN. Previously, we described that the simultaneous gradient descent often does not lead to convergence of a GAN (see section 3.4). Therefore, Arjovsky et al. (2017) propose a Wasserstein GAN (WGAN). Instead of the Jensen-Shannon divergence in a GAN proposed by Goodfellow et al. (2014a), the authors propose Earth-Mover (EM) distance to minimize the difference between p_{data} and p_G. Informally, the Earth-Mover distance measures the amount of cost, referred to as "*dirt*" it takes to transform p_G to approximate p_{data}. Intuitively, the dirt is defined as the mass of the distribution multiplied with the distance the mass needs to travel to approximate the real data distribution (Arjovsky et al. 2017).

The authors provide proof that the Wasserstein distance has the guarantee of being differentiable and continuous almost everywhere in contrast to the Kullback-Leibler divergence. To illustrate this, Arjovsky et al. (2017) propose to consider two uniform distributions. As the distance between these two distributions increases, the Kullback-Leibler divergence approaches infinity through division by zero. For example, the scenario where the two pdfs do not have overlap. This implies that we are unable to derive the gradient, because there is no loss function to minimize. See Equation 19 for a formal definition of the Kullback-Leibler divergence.

The Wasserstein distance is defined even if there is no overlap between the two distributions. Therefore, the Wasserstein distance is always differentiable and we are always able to update the weights of the generator (Arjovsky et al. 2017). Now that the Wasserstein distance is differentiable almost everywhere, D can be trained until convergence and G is always able to "*catch up*". Intuitively, an optimal D is able to give the most accurate loss function to G. Whereas, earlier we needed to account for a delicate balance between D and G. For example, to prevent mode collapse.

In contrast to Equation 1, the discriminator D acts as a *critic* instead of a *detective*. The discriminator does not longer determine whether the samples are from p_{data} or p_G with a sigmoid activation function, but whether to what degree the samples are from p_{data} or p_G with a linear activation function. Consequently, the authors empirically show that the convergence of the WGAN is more stable. During their empirical study, the authors did not find any evidence for mode collapse during training (Arjovsky et al. 2017). Furthermore, the Wasserstein distance has the ability to provide a meaningful loss metric that correlates with the quality of the artificial images (Arjovsky et al. 2017). Nonetheless, to date this property has not been investigated for low dimensional marketing data generation.

3.5.3. Wasserstein GAN with gradient penalty. Gulrajani et al. (2017) propose an alternative to the WGAN from Arjovsky et al. (2017). The authors describe that the weight clipping applied in a WGAN reduces the critic in being able to represent complex functions. In other words, the capacity of the critic is reduced. As a result, the critic is only able to represent very simple functions. Furthermore, the authors show that the combination of the weight clipping and loss function in the Wasserstein GAN leads to vanishing or exploding gradients. Gulrajani et al. (2017) introduce a WGAN with gradient penalty (WGAN-GP) to overcome these issues. Instead of reducing the capacity (weights) of the critic, the authors introduce a L^2-norm on the gradients of the critic and add the norm to the loss function of the Wasserstein GAN. Therefore, the authors do not reduce the available functions the critic can represent, but the size of the gradient steps it can take. Gulrajani et al. (2017) show empirically that this leads to a more stable convergence of GANs.

3.5.4. Convolutional networks. Radford et al. (2015) build further on the original paper of Goodfellow et al. (2014a). The authors describe the recent success of convolutional networks in image recognition networks and apply it to the architecture of a GAN (Krizhevsky et al. 2012). Theoretically, convolutional layers have several advantages compared to fully connected layers (Goodfellow et al. 2016).

First of all, the convolutional layers use a prespecified number of kernels $\boldsymbol{k}$ with the convolution operation that moves over the data (see Figure 2). The weights of the kernel are learned to detect the most informative features for subsequent layers. Therefore, we only have to store the weights of the kernel, which leads to computational and statistical efficiency. Simply put, we have fewer weights and more data. Secondly, the features are stored as activations in as many feature maps $\boldsymbol{x} * \boldsymbol{k}$ as there are kernels (Goodfellow et al. 2016).

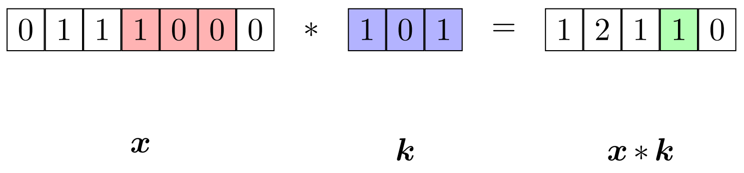

Figure 2 **Example of a 1D convolution operation. x refers to a vector of input data. Subsequently, the kernel k of size 3 moves over x with a stride of 1. The stride refers to the step size the kernel takes over the input data x. The result is as many feature maps $x * k$ as prespecified number of kernels. The scalars in the feature map $x * k$ are the activations. The activations are usually manipulated with an activation function such as ReLU, tanh or even dropout. In this example the convolution operation is as follows: 1(1) + 0(0) + 0(1) = 1. In this example, the size of the vector x shrinks. This is what we call valid padding (Goodfellow et al. 2016). To keep the same size of the vector x, one can use zero padding. Zero padding adds zeros to the vector x to ensure that the feature map $x * k$ remains of the same size.**

Therefore, convolutional layers are often referred to as feature detectors. These learned features are simple in lower layers, and become increasingly abstract in subsequent layers (Goodfellow et al. 2016). Ultimately, these features can help to explain the variance in a variable of interest. In conclusion, deep convolutional generative adversarial networks (DCGANs) offer advantages compared to fully connected GANs from Goodfellow et al. (2014a) and are heavily used in the literature to generate images (Radford et al. 2015, Beaulieu-Jones et al. 2019, Karras et al. 2017).

3.5.5. Recurrent neural networks and LSTM. In this section, we propose a recurrent GAN (RGAN) to generate longitudinal data. The literature has developed recurrent neural networks (RNNs) to learn temporal sequences (Rumelhart et al. 1986). For example, Google uses RNNs for their smart assistant Allo, Google Translate and Google Assistant (Beaufays 2015, Khaitan 2016). Moreover, Apple uses RNNs for Siri and Amazon for Amazon Alexa (Raeesy et al. 2018).

Neural networks are often referred to as feedforward neural networks as we propagate forward from the input layer to the output layer to arrive at a prediction. However, longitudinal data brings an additional time dimension over which we want to learn the relationships. Goodfellow et al. (2016) describe that recurrent neural networks include a trainable hidden state or sequence of hidden states often denoted by $\boldsymbol{h}$ at time t (see Figure 3). These hidden states could be compared to the hidden layer in a feedforward neural network only now the hidden state is applied recurrently. This creates a representation $\boldsymbol{h}$ of what should be remembered over time t. Different from a feedforward neural network, the weight matrix $\boldsymbol{W}$ is introduced and *"recurrently"* applied to learn what should be remembered over time (see left of Figure 3). In Figure 3, we unfold the computational graph of the network over time, we observe that the same weight matrix $\boldsymbol{W}$ learns a summary of what should be remembered over time. To illustrate unfolding, consider a function of the hidden state at t = 3, $\boldsymbol{h}^3 = f(\boldsymbol{h}^2; \theta)$. If we unfold the function we obtain $\boldsymbol{h}^3 = f(f(\boldsymbol{h}^1; \theta); \theta)$.

Unfortunately, recurrent neural networks suffer from vanishing or exploding gradients (Hochreiter and Schmidhuber 1997). The same weight matrix $\boldsymbol{W}$ is applied recurrently, which means that if we have a long time sequence in the data, the weight matrix is written as $\boldsymbol{W}^t$. If we consider the Hessian of the loss function with respect to the weights $\boldsymbol{W}$, the eigenvalues of the Hessian determine the optimal learning rate α (i.e., curvature). In RNNs, the eigenvalues become an exponential function of the length of the time sequence t. Therefore, the curvature increases exponentially over t and a large learning rate arises to account for the steep curvature (i.e., exploding gradients). In this way, the learning becomes very slow and unstable, because the gradient step is too large and might catapult over the desired minimum. Or if the eigenvalues are between 0 and 1, the gradient goes to zero or vanishes. Subsequently, we are unable to determine in which direction we should update the weight matrix $\boldsymbol{W}$ (Bengio et al. 1993).

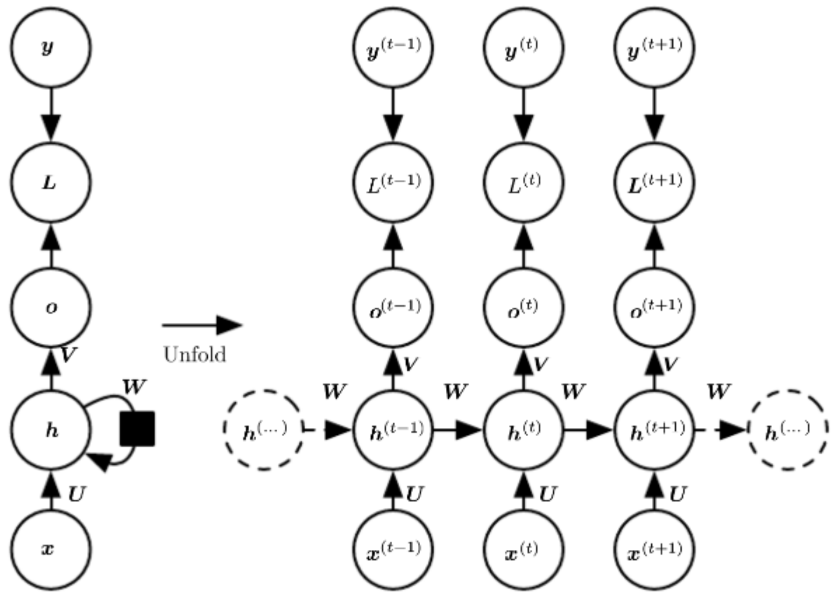

Figure 3 Architecture of a recurrent neural network. h is the hidden state that captures relationships over time t. The V, W and U represent trainable weight matrices by gradient descent updates. The weight matrix W is applied recurrently over time. o is the output that is used to calculate the loss L with the true value y at time step t. In this architecture we output a sequence, however variations of this architecture such as a final prediction at the end of t are also possible. Figure adapted from (Goodfellow et al. 2016).

These shortcomings in RNNs lead Hochreiter and Schmidhuber (1997) to develop Long Short-Term Memory (LSTM) networks. Hochreiter and Schmidhuber (1997) describes that LSTMs overcome the vanishing gradient problem by introducing an input gate, forget gate and output gate. These gates are able to transform the hidden state $\boldsymbol{h}^{(t)}$ over time. First of all, the input gate decides which values we update from $\boldsymbol{h}^{(t-1)}$. The forget gate takes the previous hidden state $\boldsymbol{h}^{(t-1)}$ and $\boldsymbol{x}$ and determines whether we should forget or keep information from the representation $\boldsymbol{h}^{(t-1)}$. Finally, the output layer decides what is transferred to the following hidden state $\boldsymbol{h}^{(t)}$. Each of these gates have a corresponding weight matrix which allows the network to learn more complicated dynamics over time. Graves et al. (2009) describe how LSTMs outperform state-of-the-art sequence prediction models. Compared to these models, LSTMs do not make a Markovian assumption which makes learning contextual effect of long sequences difficult (Graves et al. 2009).

4. Theoretical analysis

In this chapter, we introduce theoretical concepts to understand why neural networks are the appropriate method to sample privacy-preserving data. Moreover, we provide a theoretical analysis of the approximation of the data generating process p_{data} by p_G in case of a generative adversarial network (Goodfellow et al. 2014a).

4.1. Universal approximation theorem

In section 3.3, we describe that if we increase the number of hidden units, the variance of values for the loss function decreases (Choromanska et al. 2014). Therefore, a question arises of how many hidden units are appropriate for a specific task (Williams 1997). As we will describe, an infinite number of hidden units results in theoretical attractive properties for the approximation of functions for neural networks.

Leshno et al. (1993), Williams (1997) develop the theoretical notion that the neural networks are universal function approximators as the number of hidden units (width) goes to infinity. Intuitively, the proofs for such statements in the literature consists of a statement that for any bounded continuous function $f : \mathbb{R}^n \rightarrow \mathbb{R}^k$ there exists a neural network N with one hidden layer and tolerance ε such that $\|f - N\| < \varepsilon$. This implies that neural networks are able to represent *any* continuous function to an arbitrary degree of tolerance ε (Goodfellow et al. 2016). Unfortunately, such neural networks are far too large to have any practical value, because it is computationally too expensive to train such one-layer wide networks. Therefore, one could question the practical value of neural networks to represent functions such as image or customer churn recognition. Furthermore, a neural network is able to represent the function but not necessarily learn the function. For example, we might get stuck in a local minimum instead of a global minimum.

On the verge of the breakthrough of deep learning, Bengio (2009) finds that as we increase the layers (depth) of a neural network, we require less hidden units (width) to represent the function successfully. The theoretical argument is that deeper models are able to reduce the complexity of the high-dimensional vector space the observations lie in (Bengio 2009). To illustrate, the vector space grows linear with the number of variables. However, the number of unique configurations of a set of variables grows exponentially with each input variable introduced. Due to the high-dimensionality of these vector spaces, we need a data set with a considerable number of observations to learn, for example, which customers are likely to churn (i.e., the curse of dimensionality).

4.2. Manifold learning

In deep- and machine learning, we often refer to learning a manifold. Goodfellow et al. (2016) describe that although a mathematical definition of a manifold exists, machine learning usually defines it more general. To explain a manifold, we go on a small adventure into the geometric imagination. A manifold $\mathcal{M}$ is a lower-dimensional subset of observations embedded in a high-dimensional vector space. For example, we can think of roads on earth as a one-dimensional manifold in a three-dimensional universe, at least that is how humans perceive the world. Or four-dimensional universe, if we consider spacetime from Einstein.

We can express a probability distribution around the subset of observations in the high-dimensional vector space (e.g., kernel density estimation). Now consider the following case where you generate an image of 1024×1024 with 3 colour channels by randomly sampling a value for each pixel. The probability is very small that we obtain a realistic image of a human face. Therefore, the probability distribution is a highly concentrated surface in $\mathbb{R}^{1024\times1024\times3}$. With highly concentrated we mean that the probability distribution has the probability value zero for most observations.

The probability distribution is often visualized as a curved sheet in a high-dimensional vector space. We can go into tangent directions that lie on the probability distribution to arrive at other probable images. Only in a small number of directions, the probability distribution remains large. If we move orthogonal from the probability distribution, the probability quickly goes to zero. Goodfellow et al. (2016) describe that there are deep learning architectures that only learn the variations that are allowed by the manifold (i.e., autoencoders). The idea of moving orthogonal to the manifold is one of the reasons why deep learning algorithms remain vulnerable and is an entirely new research stream called adversarial machine learning (Goodfellow et al. 2014b). Finally, we can extend the idea of obtaining the image of a human face to many other real-world examples. As we describe in the following paragraphs, one of the biggest advantages of deep neural networks arises from a non-linear manipulation to the manifold of observations.

Goodfellow et al. (2016) describe how decision trees are able to represent a function that divides the high-dimensional vector space into decision regions to classify observations. Formally, a decision tree that has the objective to classify observations is a decision function $h : \mathbb{R}^n \to [0,1]^k$ that partitions the vector space $\mathbb{R}^n$ in k decision regions $\mathcal{R}_1, \ldots, \mathcal{R}_k$ (Fawzi et al. 2017). Note that these decision regions of one class do not need to be connected. When we train an algorithm such as a decision tree, we aim to learn the function that divides the vector space into optimal decision regions to classify all observations correctly. Goodfellow et al. (2016) describe that the number of decision regions is linear to the number of leaves in a decision tree. Or in k-nearest neighbours the number of regions is equal to k predefined clusters. Such algorithms are defined as local learning algorithms. To generalize to new observations, these local greedy algorithms need rely on the smoothness prior (Goodfellow et al. 2016). The prior states that the learned function around the observed data points (locally) does not change much. However, the smoothness prior is not enough to represent a complex data generating progress, because we are only able to represent a number of regions linear to the number of leaves or predefined clusters. In the next paragraph, we describe how neural networks overcome this limitation.

Compared to local learning algorithms, Bengio (2009) describe that deep neural networks with multiple layers consist of composite non-linear functions (e.g., with three layers $f(\boldsymbol{x}) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(\boldsymbol{x})\right)\right)$. The composition of non-linear functions allows the network to use representations learned in the earlier layers to carry over to higher layers in the network (Goodfellow et al. 2016). Intuitively, lower layers learn simpler representations, and higher layers build on them to learn more abstract concepts. For example, in large image recognition networks, the lower layers detect edges and lighting, similar to what the simple cells of the human primary visual cortex register (Olshausen and Field 1996). Intuitively, deep networks are able to non-linearly fold the curved manifold with the creation of these representations. It is no exception that the classes are linearly separable in the final layer of the network (Salakhutdinov and Hinton 2007). Interestingly, Hauser and Ray (2017) show visually that a 40-layer deep neural network learns representations in each layer of a neural network to make the classes linearly separable (see Figure 4). Therefore, one could interpret the first layers of deep neural networks as a non-linear feature extraction mechanisms. This allows the network to learn about relationships between the learned regions in a high-dimensional space of observations.

Finally, Bengio (2009) describes that deep neural networks are *non-local learning* algorithms which means that the number of regions are exponential in the number of hidden units. Intuitively, each neuron can contribute to multiple regions and multiple units can contribute to one region. In other words, neural networks use distributed representations (Goodfellow et al. 2016). Deep neural architectures do not only require less hidden units to represent more regions, but they also require less training observations compared to local learning algorithms such as decision trees (Bengio 2009, Goodfellow et al. 2013). Unfortunately, we here end our trip into the geometric imagination with theoretical arguments to use deep neural networks.

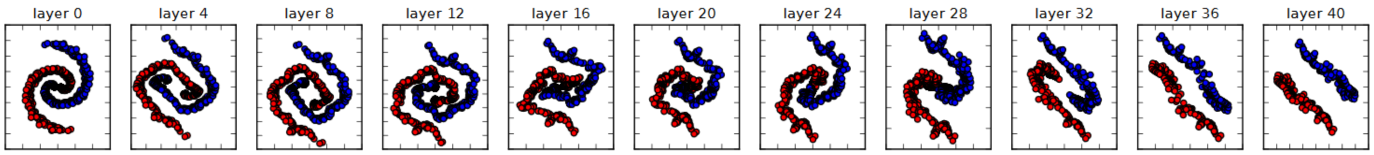

Figure 4 **Illustration from Hauser and Ray (2017). First, the authors reduced the dimensionality of the manifold to two dimensions. The neural network learns to non-linearly transform the input or manifold (layer 0) until the case where the classes (blue and red) are linearly separable (layer 40). In other words, in each layer the neural networks learns a new representation of the original input data.**

4.3. Generative adversarial networks

In this section, we aim to give more intuition into the proof from Goodfellow et al. (2014a). The authors provide proof for $p_G = p_{\text{data}}$. Recall, that we defined p_G as the distribution of the random variable G and that p_{data} is the data generation process or real distribution. In the situation of $p_G = p_{\text{data}}$, the distribution of the generator is equal to the data generating process.

Goodfellow et al. (2014a) take the value function $V(D, G)$ from Equation 1 and use the equality:

$$\mathbb{E}_{\boldsymbol{z} \sim p_Z(\boldsymbol{z})} \log(1 - D(G(\boldsymbol{z}))) = \mathbb{E}_{\boldsymbol{x} \sim p_G(\boldsymbol{x})} \log(1 - D(\boldsymbol{x})). \tag{5}$$

At first glance, one could argue that the equality in Equation 5 comes from a neural network G that applies a transformation with a unique set of parameters such that $G(\boldsymbol{z})$ leads to samples of $\boldsymbol{x}$. Subsequently, we could assume that an inverse function G^{-1} of G exists to go from $\boldsymbol{x}$ back to samples of $\boldsymbol{z}$. However, in practice a neural network need not be an invertible function. For example, if we use a ReLU activation function in the hidden layers, all the negative inputs are mapped to zero. Intuitively, consider the situation where we use a ReLU activation function and the activation is -10. The output of the activation function would be: $\max(0, -10) = 0$. Subsequently, if we adjust the weight so that the activation becomes -20, the result from the ReLU activation function is still zero $\max(0, -20) = 0$. As a result, the inverse function G^{-1} is not unique. Therefore, we argue that we are not able to take the inverse function of G.

We argue that the equality is a result from measure theory. Specifically, a Radon-Nikodym derivative from the Radon-Nikodym Theorem (Billingsley 1986, Biau et al. 2018).

We illustrate the Radon-Nikodym derivative with an intuitive example. Consider a game where we have four balls with different colors (red, green, yellow, purple). For a red ball we receive 1 euro, the green ball 2 euros, the yellow ball 3 euros and the purple ball 4 euros. For each ball there is a probability of $\frac{1}{4}$ to draw the ball from a glass jar. The expected profit under the probability measure z is calculated as follows: $\mathbb{E}_{X\sim p_z} = \sum_{i=1}^{n} x_i p_z(x_i) = 1(\frac{1}{4}) + 2(\frac{1}{4}) + 3(\frac{1}{4}) + 4(\frac{1}{4}) = 2.5$ euros. Now let us introduce new probabilities for each ball. This results in a new expected profit under the new probability measure g of $\mathbb{E}_{X\sim p_g} = \sum_{i=1}^{n} x_i p_g(x_i) = 1(\frac{2}{10}) + 2(\frac{3}{10}) + 3(\frac{4}{10}) + 4(\frac{1}{10}) = 2.4$ euros.

Now we can use the Radon-Nikodym derivative. To switch between the probability measures z and g, we only have to multiply or divide by the Radon-Nikodym derivative ξ. In this example, we can write: $\mathbb{E}_{X\sim p_g} = \mathbb{E}_{X\sim p_z}\xi$ to go from the probability mass function (pmf) p_z to the pmf p_g. We only have to multiply or divide by $\xi = \frac{2.4}{2.5}$ to go from one probability measure to another. We end our example with the main insight that we can switch between probability measures (i.e., pmfs or pdfs) with the Radon-Nikodym derivative. In a continuous case where we want to switch between pdfs, the Radon-Nikodym derivative is a function over which we can integrate (Billingsley 1986). An example of a Radon-Nikodym derivative in a continuous case is an indicator function (Billingsley 1986).

Now we can return to the equality in Equation 5 given by Goodfellow et al. (2014a). In this equality, the Radon-Nikodym theorem tells us that there exists a Radon-Nikodym derivative to arrive at:

$$\begin{aligned} V(D,G) &:= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log(D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_Z}[\log(1-D(G(\boldsymbol{z}))] \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log D(\boldsymbol{x}) \mathrm{d}x + \int_z p(\boldsymbol{z}) \log(1-D(G(\boldsymbol{z})))\mathrm{d}z \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log D(\boldsymbol{x}) + p_G(\boldsymbol{x}) \log(1-D(\boldsymbol{x}))\mathrm{d}x. \end{aligned} \tag{6}$$

Subsequently, remember that the goal of the discriminator D is to maximize Equation 6 (see Equation 1). If G is given, we can rewrite Equation 6 as: $f(y) = a\log y + b\log(1-y)$. To find the maximum of a discriminator D given a generator G, we take a first order derivative of $f(y)$ and set it equal to zero:

$$f'(y) = 0 \Rightarrow \frac{a}{y} + \frac{b}{1-y} = 0 \Rightarrow \frac{-a+ay-by}{y(y-1)} = 0 \Rightarrow -a+ay-by = 0 \Rightarrow y(a-b)-a = 0 \Rightarrow y = \frac{a}{a-b} \tag{7}$$

We can determine whether this is a maximum with $f''(y)$:

$$f''(y) = 0 \Rightarrow -\frac{a}{y^2} - \frac{b}{(1-y)^2} = 0 \Rightarrow -\frac{a}{(\frac{a}{a-b})^2} - \frac{b}{(1-\frac{a}{a-b})^2} < 0 \tag{8}$$

Thus, we can conclude that $\frac{a}{a+b}$ is indeed a maximum (i.e., $f''(y) < 0$). Goodfellow et al. (2014a) provide further evidence that the maximum $\frac{a}{a+b}$ must be a unique maximum on the domain given $a, b \in (0,1)$ and $a + b \neq 0$. Therefore, we find that the optimal discriminator given G is:

$$D^*_G(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})} \text{ and } 1 - D^*_G(\boldsymbol{x}) \quad = \frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})}. \tag{9}$$

Goodfellow et al. (2014a) describes that with the definition of an optimal discriminator we can reformulate the value function from Equation 6 and define a virtual training criteria for the generator $C(G)$:

$$C(G) = \max_D V(D^*_G, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})}] + \mathbb{E}_{\boldsymbol{x} \sim p_G}[\log \frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})}]. \tag{10}$$

Now that we have the optimal discriminator D for a given generator G, we have to find a global minimum of G. Goodfellow et al. (2014a) claim that the global minimum of $C(G)$ is achieved if and only if $p_G = p_{\text{data}}$. In the first direction, given that $p_{\text{data}} = p_G$, we arrive at the optimal discriminator that is unable to distinguish real from artificial samples:

$$D^*_G(\boldsymbol{x}) = \frac{1}{2} \text{ and } 1 - D^*_G(\boldsymbol{x}) = \frac{1}{2}. \tag{11}$$

This represents the scenario where the discriminator is unable to distinguish between samples from p_{data} and p_G. Subsequently, Goodfellow et al. (2014a) plugs the optimal discriminator $D^*_G(\boldsymbol{x})$ back into the value function from Equation 6 to obtain a candidate value for a global minimum:

$$\begin{aligned} C(G) :&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g} [\log (1 - D^*_G(\boldsymbol{x}))] \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(\frac{1}{2}) + p_G(\boldsymbol{x}) \log(\frac{1}{2}) \mathrm{d}x. \end{aligned} \tag{12}$$

Subsequently, we can integrate over the entire domain of both $p_{\text{data}}(\boldsymbol{x})$ and $p_G(\boldsymbol{x})$ with respect to x. The integrals of both pdfs are by definition equal to one. Such that:

$$= \log \frac{1}{2} + \log \frac{1}{2} = -\log 4. \tag{13}$$

The value $-\log 4$ is a candidate value for the global minimum. Now we want to prove that this is a unique minimum for the generator. Therefore, we drop the assumption $p_G = p_{\text{data}}$ for now and observe that for any G, we can plug in D_G^* into the equation where the discriminator achieves its maximum (see Equation 12):

$$\begin{aligned} C(G) &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})}] + \mathbb{E}_{\boldsymbol{x}\sim p_G}[\log \frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})}] \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log[\frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})}] + p_G(\boldsymbol{x})[\log \frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})}]\mathrm{d}x. \end{aligned} \tag{14}$$

Subsequently, we use a trick to add and subtract $\log 2$ and multiply with a probability distribution in Equation 14, which is equal to adding zero to both integrals (Rome 2017):

$$\begin{aligned} C(G) = \int_{\boldsymbol{x}} &(\log 2 - \log 2) p_{\text{data}}(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x}) \log \left(\frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})} \right) \\ &+ (\log 2 - \log 2) p_G(\boldsymbol{x}) + p_G(\boldsymbol{x}) \log \left(\frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right) \mathrm{d}x \end{aligned} \tag{15}$$

Subsequently, we can rewrite:

$$\begin{aligned} &= \int_{\boldsymbol{x}} \log 2 p_{\text{data}}(\boldsymbol{x}) - \log 2 p_{\text{data}}(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x}) \log \left(\frac{p_{\text{data}}(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right) \\ &+ \log 2 p_G(\boldsymbol{x}) - \log 2 p_G(\boldsymbol{x}) + p_G(\boldsymbol{x}) \log \left(\frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right) \mathrm{d}x \\ &= \int_{\boldsymbol{x}} -\log 2 p_{\text{data}}(\boldsymbol{x}) - \log 2 p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x}) \log \left(\frac{p_{\text{data}}(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right) \\ &+ \log 2 p_{\text{data}}(\boldsymbol{x}) + \log 2 p_G(\boldsymbol{x}) + p_G(\boldsymbol{x}) \log \left(\frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right) \mathrm{d}x \\ &= \int_{\boldsymbol{x}} -\log 2 (p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})) + p_{\text{data}}(\boldsymbol{x}) \log \left(\frac{p_{\text{data}}(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right) \\ &+ \log 2 p_{\text{data}}(\boldsymbol{x}) + \log 2 p_G(\boldsymbol{x}) + p_G(\boldsymbol{x}) \log \left(\frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right) \mathrm{d}x \end{aligned} \tag{16}$$

Eventually, we can integrate $p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})$ over x and use linearity of the integral to rewrite as:

$$
\begin{aligned}
&= -\log 4 + \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log \left(\frac{p_{\text{data}}(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right) \\
&+ \log 2p_{\text{data}}(\boldsymbol{x}) + \log 2p_G(\boldsymbol{x}) + p_G(\boldsymbol{x}) \log \left(\frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right) \mathrm{d}x \\
&= -\log 4 + \int_{\boldsymbol{x}} \log 2p_{\text{data}}(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x}) \log \left(\frac{p_{\text{data}}(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right) \\
&+ \log 2p_G(\boldsymbol{x}) + p_G(\boldsymbol{x}) \log \left(\frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right) \mathrm{d}x \\
&= -\log 4 + \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) (\log 2 + \log \left(\frac{p_{\text{data}}(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right)) \\
&+ p_G(\boldsymbol{x}) (\log 2 + \log \left(\frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right)) \mathrm{d}x
\end{aligned} \tag{17}
$$

Now, we can use the logarithmic product rule for $\log 2 + \log \left(\frac{p_{\text{data}}(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right)$ and $\log 2 + \log \left(\frac{p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right)$ to arrive at:

$$
\begin{aligned}
&= -\log 4 + \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) (\log \left(\frac{2p_{\text{data}}(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right)) \\
&+ p_G(\boldsymbol{x}) (\log \left(\frac{2p_G(\boldsymbol{x})}{p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})} \right)) \mathrm{d}x \\
&= -\log 4 + \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) (\log \left(\frac{p_{\text{data}}(\boldsymbol{x})}{(p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x}))/2} \right)) \\
&+ p_G(\boldsymbol{x}) (\log \left(\frac{p_G(\boldsymbol{x})}{(p_G(\boldsymbol{x}) + p_{\text{data}}(\boldsymbol{x})/2} \right)) \mathrm{d}x
\end{aligned} \tag{18}
$$

Subsequently, Goodfellow et al. (2014a) largely draw from information theory and use a definition of the Kullback-Leibler and Jensen-Shannon divergence to show that $-\log 4$ is a unique global minimum. The Kullback-Leibler divergence for probability measures P and Q of a continuous random variable is defined as follows (Bishop 2006):

$$
\mathrm{KL}(P \| Q) = \int_{\boldsymbol{x}} p(\boldsymbol{x}) \log \left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \right) \mathrm{d}x \tag{19}
$$

Intuitively, the Kullback-Leibler divergence measures the difference between two probability distributions. We arrive at Equation 5 from the paper of Goodfellow et al. (2014a), where the authors apply the definition of the Kullback-Leibler divergence in Equation 18 to arrive at:

$$
C(G) = -\log 4 + \mathrm{KL} \left(p_{\text{data}}(\boldsymbol{x}) \| \frac{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})}{2} \right) + \mathrm{KL} \left(p_G(\boldsymbol{x}) \| \frac{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})}{2} \right) \tag{20}
$$

Bishop (2006) shows that with Jensen's inequality for a convex function and random variable X: $\mathrm{E}[f(X)] \geqslant f(\mathrm{E}[X])$ and the fact that $f(x) = -\ln x$ is a strictly convex function, that the Kullback-Leibler divergence non-negative is if and only if $p(\boldsymbol{x}) = q(\boldsymbol{x})$ for all $\boldsymbol{x}$. Therefore, we take the definition of the Kullback-Leibler divergence from Equation 19 and use the logarithm quotient rule $\log(\frac{z}{x}) = -\log(\frac{x}{z})$ to arrive at:

$$\mathrm{KL}(P\|Q) = -\int_{\boldsymbol{x}} p(\boldsymbol{x}) \log\left(\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right) \mathrm{d}x \tag{21}$$

Next, we use Jensen's inequality to prove that the Kullback-Leibler divergence from Equation 21 has to be greater or equal to zero:

$$\begin{aligned}
\mathrm{KL}(P\|Q) &= -\int_{\boldsymbol{x}} p(\boldsymbol{x}) \log\left(\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right) \mathrm{d}x \\
&= -\int_{\boldsymbol{x}} p(\boldsymbol{x}) \log\left(\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right) \mathrm{d}x \geqslant -\log\left(\int_{\boldsymbol{x}} p(\boldsymbol{x}) \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right) \mathrm{d}x \\
&= -\int_{\boldsymbol{x}} p(\boldsymbol{x}) \log\left(\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right) \mathrm{d}x \geqslant -\log\left(\int_{\boldsymbol{x}} \frac{p(\boldsymbol{x}) q(\boldsymbol{x})}{p(\boldsymbol{x})}\right) \mathrm{d}x \\
&= -\int_{\boldsymbol{x}} p(\boldsymbol{x}) \log\left(\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right) \mathrm{d}x \geqslant -\log\left(\int_{\boldsymbol{x}} q(\boldsymbol{x})\right) \mathrm{d}x \\
&= -\int_{\boldsymbol{x}} p(\boldsymbol{x}) \log\left(\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right) \mathrm{d}x \geqslant 0
\end{aligned} \tag{22}$$

Or alternatively using $-\log\left(\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right) = \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right)$:

$$= \int p(\boldsymbol{x}) \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) \mathrm{d}x \geqslant 0 \tag{23}$$

Finally, we use the result from Equation 23 to show that the Kullback-Leibler divergence must be equal or bigger than zero in Equation 20. This shows that the global minimum must be $-\log 4$. Finally, Goodfellow et al. (2014a) use the definition of the Jensen-Shannon divergence in Equation 20 to prove that only one G is able to achieve this minimum (Lin 1991):

$$\mathrm{JSD}(P\|Q) = \frac{1}{2}\mathrm{KL}(P\|(P+Q)/2) + \frac{1}{2}\mathrm{KL}(Q\|(P+Q)/2) \tag{24}$$

If we use the definition of the Jensen-Shannon divergence for Equation 20 where $P = p_{\text{data}}$ and $Q = p_G$:

$$
\begin{aligned}
C(G) &= -\log 4 + \mathrm{KL}\left(p_{\text{data}}(\boldsymbol{x}) \| \frac{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})}{2}\right) \\
&\quad + \mathrm{KL}\left(p_G(\boldsymbol{x}) \| \frac{p_{\text{data}}(\boldsymbol{x}) + p_G(\boldsymbol{x})}{2}\right) \\
&= -\log 4 + 2 \cdot \mathrm{JSD}\left(p_{\text{data}}(\boldsymbol{x}) \| p_G(\boldsymbol{x})\right)
\end{aligned} \tag{25}
$$

We show that the Kullback-Leibler divergence must be equal or greater than zero, so we can extend this idea to the Jensen-Shannon divergence (Lin 1991). The Jensen-Shannon divergence between two distributions is always non-negative and zero if and only if $p_G = p_{\text{data}}$ for any value of $\boldsymbol{x}$ (Goodfellow et al. 2014a). In conclusion, we show that $-\log 4$ is a unique global minimum of G.

Goodfellow et al. (2014a) provides evidence for the convergence of Algorithm 1. The authors describe that for each gradient step update for p_G with an infinite capacity D given G leads to a convergence of p_G into p_{data}, as for the generator a unique global minimum exists ($-\log 4$). However, in practice, we know that the optimization problem is non-convex, due to the non-linearity in the neural network. Therefore, we might never arrive at such a global minimum for both players in practice (see section 3.4). Therefore, we empirically investigate the convergence of a GAN in case of marketing data.

To empirically investigate the proof, we take a univariate perspective to the comparison of the multivariate probability distribution functions p_G and p_{data}. First, we compare the distributions and correlation for each variable from p_G and p_{data}. Secondly, we investigate whether artificial data are able to maintain the ability to derive meaningful insights from artificial data (Wedel and Kannan 2016, Wieringa et al. 2019, Rust 2019). For the artificial data to be practically useful for marketing, we are interested in whether the artificial data are able to predict marketing events, such as customer churn or supermarket sales. Furthermore, in a case where the parameter coefficients are equal for an estimation on the real and artificial data, academics are able to share the data to increase the generalizability of studies. Thereby, we aim to introduce the potential of a GAN within the current literature of marketing.

5. Empirical analysis

To evaluate our research question empirically, we consider three data sets, each with different characteristics. We use a publicly available churn data set to allow replication of our results. A real churn data set provides a realistic assessment of the ability to generate churn data and estimate models on artificial data to predict real-life events. The real churn data contains issues such as class imbalance, while a panel data set introduces the challenge to generate longitudinal data over multiple years and supermarkets. The first data set we describe in more detail is a public churn data set of 3,333 observations, which is freely available on the internet[1] (see Table 2).

Table 2 The variables available in the publicly available churn data set.

Variable	Scale	Description
Account Length	Ratio	The number of months a customer has a contract.
International Plan	Nominal	Whether the customer has an international plan.
Voicemail Plan	Nominal	Whether the customer has a voicemail plan.
Voicemail Message	Integer	The number of voicemail messages.
Day Min.	Continuous	The number of minutes called during the daytime.
Day Calls	Integer	The number of calls during the daytime.
Day Charge	Continuous	The amount charged during the daytime.
Eve Min	Continuous	The number of minutes called during the evening.
Eve Calls	Integer	The number of calls during the evening.
Eve Charge	Continuous	The amount charged during the evening.
Night Mins	Continuous	The number of minutes called during the evening.
Night Calls	Integer	The number of calls during the evening.
Night Charge	Continuous	The amount charged during the evening.
Int. Min.	Continuous	The number of international minutes called.
Int. Calls	Integer	The number of international calls.
Int. Charge	Continuous	The amount charged from international calls.
Cust. Serv. Calls	Integer	The amount of customer service calls.
Churn	Nominal	Whether the customer churned.

The second real churn data set is provided by an insurance provider in the Netherlands, which consists of 1,262,423 observations. The variables that are present in the data are available in Table 3.

The market panel data set with 4,858 observations provided by six different supermarket chains in the Netherlands over the years 2013 - 2016. The variables that are present in the data set are presented in Table 4.

5.1. Feature scaling

We scale the variables $\boldsymbol{x}$ from the real data distribution p_{data} that we feed to the GAN in the scale of $(-1, 1)$ to match the scale of the generator distribution p_G (LeCun et al. 1998, Goodfellow et al. 2016). Intuitively this makes sense, as the generator employs a tanh

[1] The data set is available at https://github.com/GilianPonte/GAN.

Table 3 The variables available in the real churn data set.

Variable	Scale	Description
Churn	Nominal	Whether the customer cancelled the contract.
Gender	Nominal	Male or female.
Age	Continuous	The number of years a customer has lived.
Rel. duration	Continuous	The duration of the relationship.
Collective	Nominal	Whether a customer is part of an insurance collective.
Size of Policy	Categorical	The size of the policy.
AV 2011	Categorical	Additional insurance package of the customer.
Complaints	Categorical	The number of complaints.
Contact	Integer	The number of contacts the customer made.
Distance to store	Categorical	The distance to a store.
Address size	Categorical	The size of the house of a customer.
Incoming contacts	Nominal	Whether the customer contacted the insurance.
# incoming contacts	Integer	The number of incoming contacts of the customer to insurance.
AV cancellation	Nominal	Whether a customer has cancelled the additional insurance.
Defaulter	Nominal	Whether somebody had trouble paying in the past.
Urbanity	Categorical	The urbanity of where the customer lives.
Social class	Categorical	The social class of the customer.
Stage of life	Categorical	The stage of life of a customer.
Income	Categorical	The income that a customer receives.
Education	Categorical	The level of education a customer received.
"BSR.groen"	Nominal	An additional insurance package.
"BSR.rood"	Nominal	An additional insurance package.
Without children	Nominal	Whether a customer had children.
Payment method	Nominal	The payment method of a customer.
Declared	Nominal	Whether a customer has declared any value.
Declared approved	Integer	How many declarations were approved by the insurance.
Declaration amount	Continuous	The amount of declarations in euros.

Table 4 The variables available in the market panel data set.

Variable	Scale	Description
Date	Categorical	The date at the time of sales.
Year	Categorical	The year at the time of sales.
Quarter	Integer	The year at the time of sales.
Week	Integer	The week at the time of sales.
Chain	Categorical	The supermarket chain.
Brand	Categorical	The brand of lemonade.
Unit Sales	Integer	The units of lemonade sold.
Price PU	Continuous	The price of the lemonade with the promotion.
BasePrice PU	Continuous	The price of the lemonade without promotion.
FeatDispl	Integer	% of stores with feature and display promotion.
DispOnly	Integer	% of stores with display promotion.
FeatOnly	Integer	% of stores with feature promotion.
Promotion	Continuous	% of discount.
Revenue	Continuous	The amount of revenue in euros.
MinTemp	Continuous	The minimum temperature in Celsius at De Bilt.
MaxTemp	Continuous	The maximum temperature in Celsius at De Bilt.
Sunshine	Continuous	The minimum temperature in De Bilt.
Rain	Continuous	Duration of rain in .1 hour.
KarvanC. Go	Continuous	Google Trends index (0-100).

activation function as output to generate data in our experiments. We scale each column vector $\boldsymbol{x}_j$ from the data set or matrix $\boldsymbol{X}_{i,j} = (x_j, \ldots, x_J)$ as follows:

$$\boldsymbol{x}_j^{(scaled)} = 2\frac{\boldsymbol{x}_j - \min(\boldsymbol{x}_j)}{\max(\boldsymbol{x}_j) - \min(\boldsymbol{x}_j)} - 1 \tag{26}$$

To obtain the original scale for marketing modeling, it follows that we rescale variables to the original scale of p_{data} using the following transformation:

$$\boldsymbol{x}_j = \frac{(\boldsymbol{x}_j^{(scaled)} + 1)(\max(\boldsymbol{x}_j) - \min(\boldsymbol{x}_j))}{2} + \min(\boldsymbol{x}_j) \tag{27}$$

Consequently, we are able to interpret samples from p_G and the data are ready to be used in marketing applications.

5.2. DCGAN architecture

The architecture of the generator and discriminator are displayed in Figure 5 and 6. Both are deep convolutional neural networks constructed of one input layer, three hidden layers and one output layer with the tanh or sigmoid activation function. The specification of these networks are derived from the literature from (Ioffe and Szegedy 2015, Radford et al. 2015, Goodfellow 2016, Goodfellow et al. 2016, Szegedy et al. 2016).

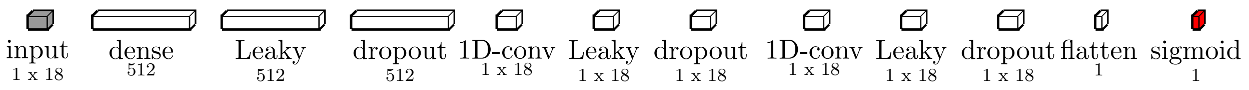

Figure 5 **Topology of the discriminator (D). Sizes at the layers refer to output shape of the layer. We use the sigmoid activation function, Adam optimizer with $\beta_1 = .5$ and $\beta_2 = .99$, Leaky ReLU ($\alpha = .2$) as hidden activation functions, dropout (40 percent) and negative log-likelihood as loss.**

5.2.1. Discriminator.

5.2.1. Discriminator. For the discriminator in Figure 5, the input layer activations take the values from a randomly sampled (with replacement) input vector $\boldsymbol{x}$ from p_{data}. We refer to the size of the layer as 1×18, because we calculate the gradient for each observation in a mini-batch and average the gradient for the mini-batch to do a weight update (see Algorithm 1). The 18 in the size of the layer refers to the dimensions of the input data. In

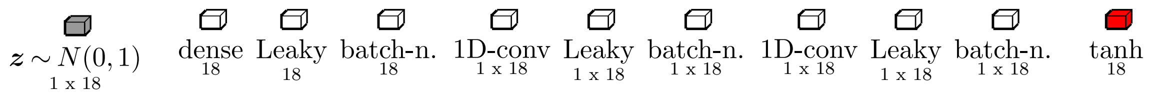

Figure 6 Topology of the generator (G). Sizes at the layers refer to output shape of the layer. We use the tanh activation function, Adam optimizer with $\beta_1 = .5$ and $\beta_2 = .99$, Leaky ReLU ($\alpha = .2$) as hidden activation functions and negative log-likelihood as loss.

case of the publicly available churn data set is 18 (see the number of variables in Table 2). Naturally, we can change the dimensionality of the first layer for the other data sets.

The input layer is connected to the first hidden dense layer with 512 units (see *dense* in Figure 5). A dense layer refers to a fully connected layer. In other words, all the activations from the input layer are connected to all the activations in the dense layer with a weight. Subsequently, we use a leaky activation function and dropout to the layer to arrive at the final activations of the dense layer. These two operations are depicted as a layer because we can clearly show where the operations occur. Moreover, one could view both operations as two layers that further manipulate the activations.

In the second hidden layer (see *1D-conv* in Figure 5), the final activations in the dense layer are connected to a one-dimensional convolutional layer that uses 18 kernels to find meaningful features in the activations of the dense layer. The discovered features are stored in the 18 feature maps (Goodfellow et al. 2016). Subsequently, we apply the Leaky ReLU activation function and dropout on the activations in the feature maps. The third layer uses the same operations as the second hidden layer. The output layer flattens the feature maps and uses a sigmoidal activation function to arrive at a prediction whether the sample is from p_G or p_{data} in a range of $(0, 1)$.

5.2.2. Generator. The generator is displayed in Figure 6. The input layer consists of a vector $\boldsymbol{z}$ randomly sampled with replacement from p_Z. The first hidden layer is a dense layer which activations are manipulated with the Leaky ReLU activation function and batch-normalization (see *batch-n.* in Figure 6). The second hidden layer is a one-dimensional convolutional layer (similar to in the discriminator) to which we apply the Leaky ReLU activation function and batch normalization. The third hidden layer is equal to the second hidden layer. Finally, we output a vector of activations in the range of $(-1, 1)$ with the tanh activation function.

5.3. Training procedures

In the case of the public churn data set, we train the DCGAN for 50,000 iterations. For each iteration both the discriminator and generator update the weights ($k = 1$, see Algorithm 1). Next, the training procedure randomly samples mini-batch of 256 observations consisting of equal variables as dimensions from p_{data}. To update the weights, we calculate the gradient for one observation at the time and average the gradients. The mini-batch from p_{data} is combined with a mini-batch from p_G over the same dimensions. This results in a sample with 512 observations. The observations from p_{data} are labelled as 1 and the observations from p_G with 0.

The DCGAN architecture uses label smoothing to induce uncertainty for the discriminator (Salimans et al. 2016). The architecture employs the optimizer Adam from Algorithm 1 in the generator and discriminator with a learning rate of .0001 and the forgetting factors $\beta_1 = .5$ and $\beta_2 = .999$ (Kingma and Ba 2014, Radford et al. 2015). The discriminator is trained on the combined sample to separate the artificial from real observations (see Algorithm 1). Consequently, the algorithm draws another mini-batch sample from p_G. This sample is labelled as 1, indicating that these observations are real. These misleading observations are fed to the DCGAN where the weights of the discriminator are frozen. In this way, the weights of the generator are updated to be able to fool the frozen discriminator into believing the sample from p_G is actually from p_{data}. We freeze the discriminator to let the generator to be able to catch up. Would the discriminator have trainable weights, then the weights of the discriminator are updated simultaneously to the generator. This would prevent the minimax game from converging. Finally, when the minimax game converges, the generator is able to fool the discriminator and generate realistic artificial observations.

5.3.1. Public churn data set. The training procedure leads to a relatively stable convergence of the public churn data set, as visible in Figure 7. One can interpret these plots as a proxy for the certainty or confidence of both players. The red line represents the generator loss and the black line represents the discriminator loss. Recall, that the discriminator has the objective to maximize, while the generator has the objective to minimize (see Equation 1). During the initial phase of training the DCGAN, the generator and discriminator display a high variance in the log-likelihood in the first ∼5,000 epochs followed by a stable convergence of both losses.

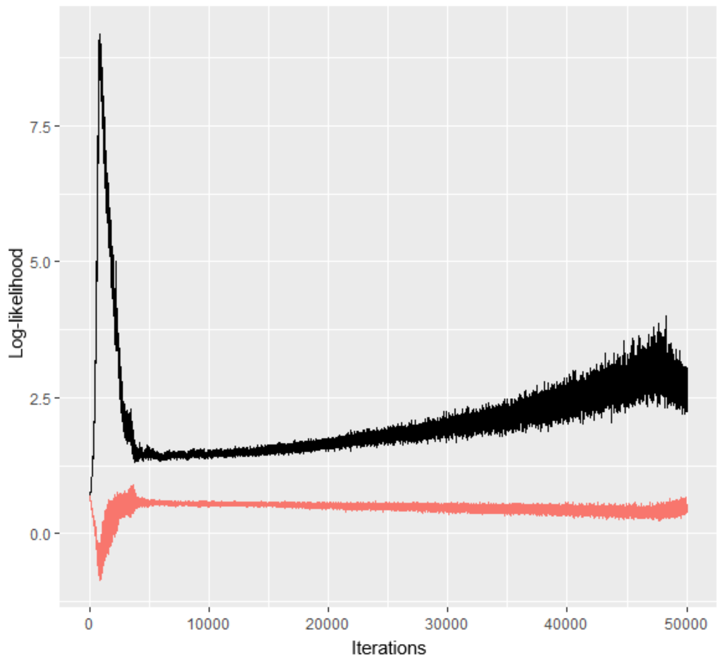

Figure 7 **The convergence of the public churn data set by the generative adversarial network (N = 50,000). The red line displays the generator loss, while the black line shows the discriminator loss.**

Visually the DCGAN is able to resemble the distributions with quite some accuracy, as displayed in Figure 8. As described in section 3.4, the increase of both losses is not necessarily an indication of the quality of the similarity of distributions. Therefore, the quality of the artificial samples is visually examined by plotting the real and artificial distribution every 1,000 iterations. The variables minutes called in the evening (*EveMins*) and minutes called internationally (*IntlMins*) are "cherry-picked" from the real and artificial data set.

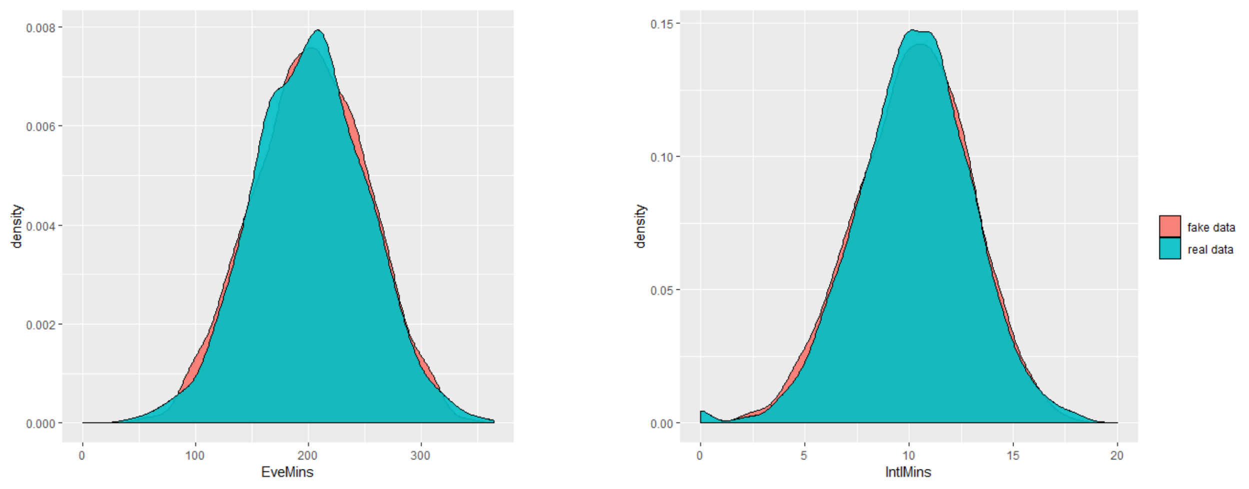

Figure 8 **Some "cherry-picked" examples of the artificial data distribution (red) and the real data distribution (blue).**

5.3.2. Real churn data set. For the real churn data set, we present the convergence of the training procedure in Figure 9. The variance over training iterations is considerably higher compared to the public churn data set in Figure 7. To illustrate the convergence of the DCGAN, a "cherry-picked" example of a comparison between the real and the corresponding artificial data distribution is displayed in Figure 9. The DCGAN is clearly having difficulties with pushing up the probabilities to estimate the extremes in the real density function. As a result, the distributions are having clear dissimilarities.

Theoretically, we consider the possibility that the discriminator lacks the capacity to approximate the distribution (Goodfellow et al. 2014a). Therefore, we conduct several experiments with varying sizes of the generator and discriminator. These experiments lead to a more stable convergence with an increase of trainable parameters for the generator to 25,860 and the discriminator to 68,166 parameters (see Figure 10). One can question whether the increase in sizes of both players in the DCGAN lead to a more accurate estimate of the real data density function. This confirms the notion of Theis et al. (2015) that a high log-likelihood does not guarantee that samples from the generator are of high-quality, as low log-likelihood does not guarantee that the samples are of low quality (see section 3.4).

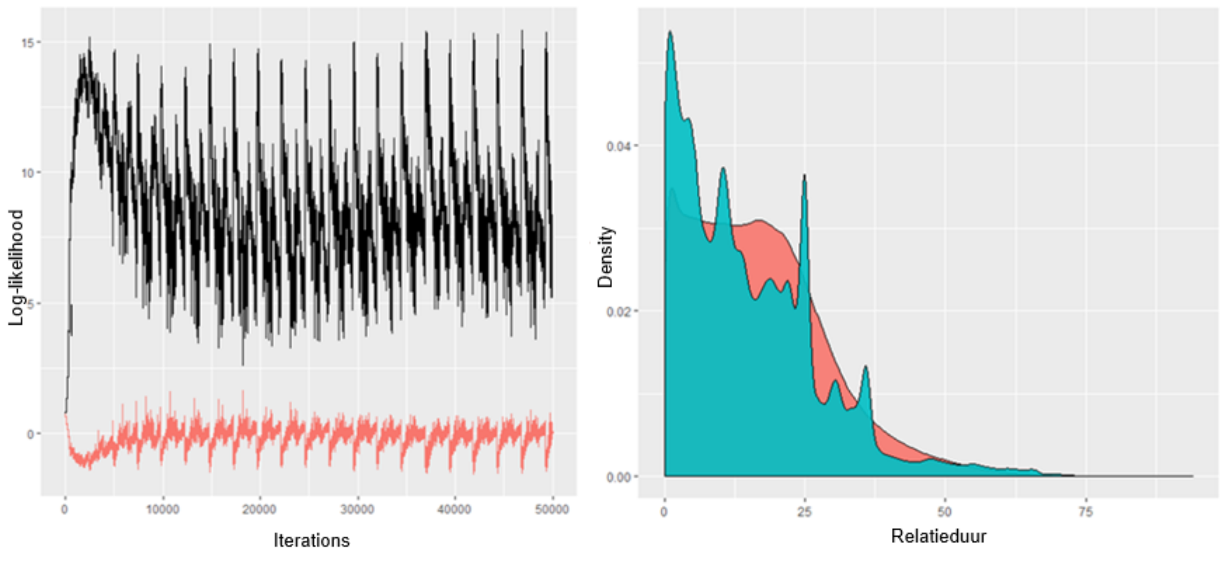

Figure 9 **The convergence of the real data set by the generative adversarial network. The artificial data distribution (red) and the real data distribution (blue) (N = 50,000). The red line displays the generator loss, while the black line shows the discriminator loss.**

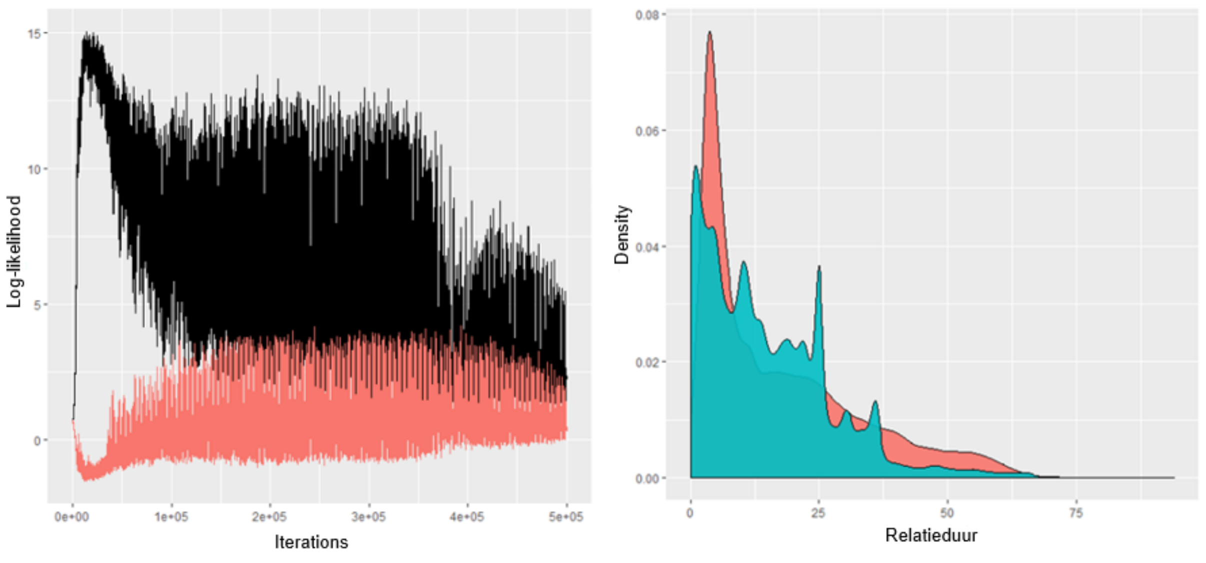

Figure 10 **An attempt of convergence of the real churn data set. The artificial data distribution (red) and the real data distribution (blue) (N = 50,000). The red line displays the generator loss, while the black line shows the discriminator loss.**

5.3.3. Lemonade sales data set. Training a DCGAN in case of the lemonade sales data set comes with more issues. The data set consists of variables with a low activity, which makes it hard for the DCGAN to generate a similar distribution. Figure 11 shows how the generator and discriminator converge to equilibrium before eventually steadily increasing. One can interpret this phenomenon as the situation where the discriminator becomes highly uncertain after a number of iterations. This is a finding in line with results from Beaulieu-Jones et al. (2019) and could indicate that the generator creates realistic examples. Our suspicion is confirmed when we compare the distributions in Figure 11. It is clear that the DCGAN is able to resemble the real data distributions.

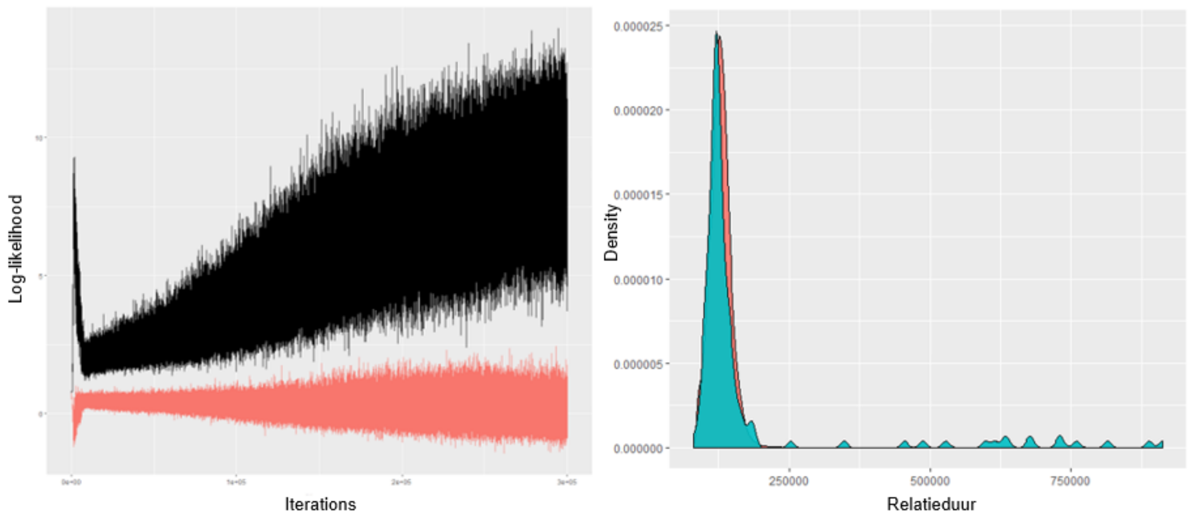

Figure 11 **The convergence of the lemonade sales data set DCGAN. The artificial data distribution (red) and the real data distribution (blue) (N = 50,000). The red line displays the generator loss, while the black line shows the discriminator loss.**

The first attempt is to generate the complete panel data set ($n = 4{,}854$). However, the DCGAN generates a multitude of observations per unique date. This made it impossible to combine two unique dates with observations of the competitors from a specific supermarket brand. Subsequently, we train the DCGAN on data from one supermarket chain ($n = 169$), before and after recoding the variables to account for multicollinearity (Leeflang et al. 2015). 169 observations are considered to be a very limited sample to successfully generate artificial data. To illustrate, similar studies have reported training a DCGAN a multitude of observations. For example, Beaulieu-Jones et al. (2019) trained a DCGAN on 9,361 observations and Kumar et al. (2018) used 5 million observations.

The limited sample size led to another issue, the DCGAN showed symptoms of mode collapse on the variables with relatively low activity (i.e., the DCGAN only generated one value for some of the variables). Therefore, a second attempt of training the DCGAN with 1,000,000 iterations is performed on the data set. The mini-batch size is decreased to create a regularization effect on the data, because a smaller mini-batch size leads to a more noisy estimate of the expected gradient (Wilson and Martinez 2003). We motivated the introduction of mini-batches in section 3.2.1. Wilson and Martinez (2003) describe that decreasing the size of a mini-batch adds noise to the learning process and the training takes longer to converge. Therefore, we expect more variability in the artificial data and reduced mode collapse. After training, the DCGAN generated 20,000 observations to prevent the generation of variables with signs of mode collapse and ensure variation in the artificial data. This artificial data set is used in subsequent analyses.

6. Results

In this chapter, we describe the results from the experiments. We provide an analysis of the relationships between the variables, the predictive validity of the artificial data, the parameters of the estimations on artificial data, compare GAN architectures and perform robustness checks.

6.1. The correlations of variables from p_G significantly correlates with p_{data}

A property of the generator is that a sample from p_G is in multitudes of the batch size. Therefore, a sample from p_G never has the exact same amount of observations as a sample from p_{data}. Therefore, we draw a random sample equal to the number of real observations from p_G. We use a χ^2 test of independence for dummy variables in Table 6 and Pearson's correlation test for continuous variables in Table 5. When the GAN is able to anonymize the data, the variables from p_G and p_{data} should have no correlation. From Table 5, it is visible that the variables from p_{data} and p_G are uncorrelated. From 6, we observe that the dummy variables from both samples are independent.

6.1.1. Public churn data set. In Figure 12, the correlations among the variables in the public churn data set are displayed. At first glance, the correlations among the variables appear to be similar in the samples from both distributions p_G and p_{data}.

Table 5 Pearson's correlation r between variables from p_G and p_{data}. In this table we show r for all the variables.

Public churn		Real churn data set		Lemonade sales data set	
Account Length	-.01	Gender	.15	Unit Sales	-.02
Voicemail Message	.01	Age	$< .00$	Price PU	-.01
Day Min.	-.02	Rel. duration	$< .00$	BasePrice PU	-.01
Day Calls	.27	Declaration amount	.00	FeatDispl	-.01
Day Charge	-.02			DispOnly	.01
Eve Min	.03			FeatOnly	-.02
Eve Calls	.01			Promotion	-.01
Eve Charge	.03			Revenue	-.02
Night Mins	-.004			MinTemp	.01
Night Calls	-.04			MaxTemp	-.01
Night Charge	-.001			Sunshine	.001
Int. Min.	.02			Rain	.02
Int. Calls	-.02			KarvamC. Go	-.01
Int. Charge	.02				
Cust. Serv. Calls	-.02				

Table 6 χ^2 test of independence for p_G and p_{data}. In this table we show χ^2 value for all the variables

Public churn data set		Real churn data set		Lemonade sales data set	
Intl. Plan	5.6	Churn	.13	Urbanity	$< .00$
Voicemail Plan	.86	Collective	1.13	Social class	$< .00$
Churn	3.1	Size of Policy	$< .00$	Stage of life	$< .00$
		AV 2011	$< .00$	Income	$< .00$
		Complaints	$< .00$	Education	$< .00$
		Contact	$< .00$	BSR.groen	2.27
		Distance to store	$< .00$	BSR.rood	.82
		Address size	$< .00$	Without children	.53
		Incoming contacts	1.06	Payment method	.10
		# incoming contacts	$< .00$	Declared	.51
		AV cancellation	.43	Declared approved	$< .00$
		Defaulter	.71		
		Urbanity	$< .00$		

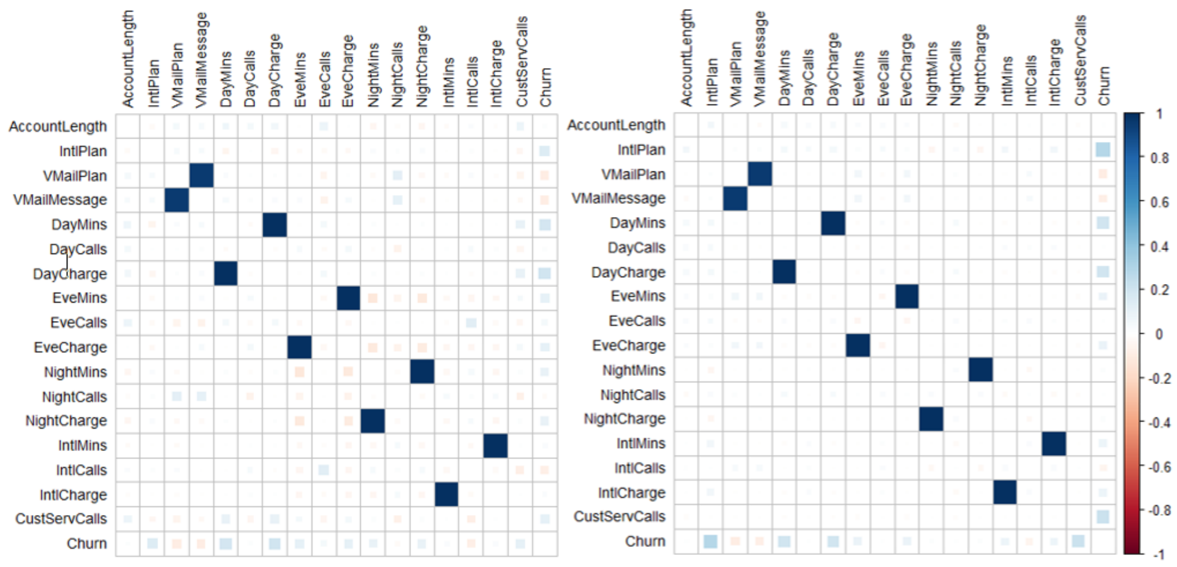

Figure 12 Correlations among a sample from p_G (left), correlations among a sample from p_{data} (right).

To formally test whether the correlation matrices are related, we convert the correlation matrices to two vectors and match the variables from both samples. The correlation of the variables tested by a Pearson's correlation test. The result indicates a statistically significant high correlation between two vectors ($r = .99$, $p < .00$). This indicates that the relationships between variables from p_{data} and p_G are highly correlated. This implies that the DCGAN is able to contain the relationships from p_{data} in p_G.

6.1.2. Real churn data set. A similar procedure leads to the creation of two vectors, where both vectors contain the correlations among p_G and p_{data}. The DCGAN is able to approximate the relationships in p_{data} with p_G with acceptable accuracy, as the correlation the relationships is high and significant ($r = .89$, $p < .00$). Figure 13 presents two data sets with similar correlations. Similarly to the public churn data set, the DCGAN is able to contain the relationships among variables.

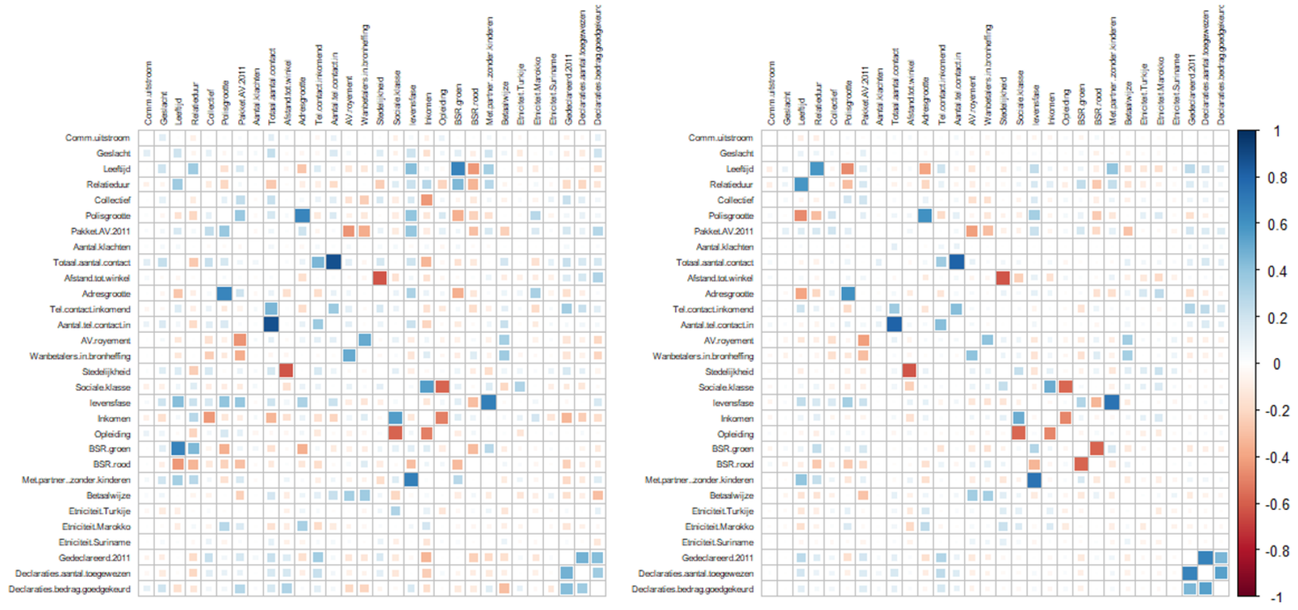

Figure 13 **Correlations among artificial data (left), correlations among real data (right).**

6.1.3. Lemonade sales data set. In Figure 14, we present the correlation among the variables from p_{data} and p_G for the lemonade sales data set. At first glance, there are some differences between the two correlation matrices. The artificial data seems to have higher correlations among variables. Whereas, the real data has lower correlations among the variables. A Pearson's correlation test indicates that the correlations between variables in the artificial data and real data are highly related ($r = .93$, $p < .00$).

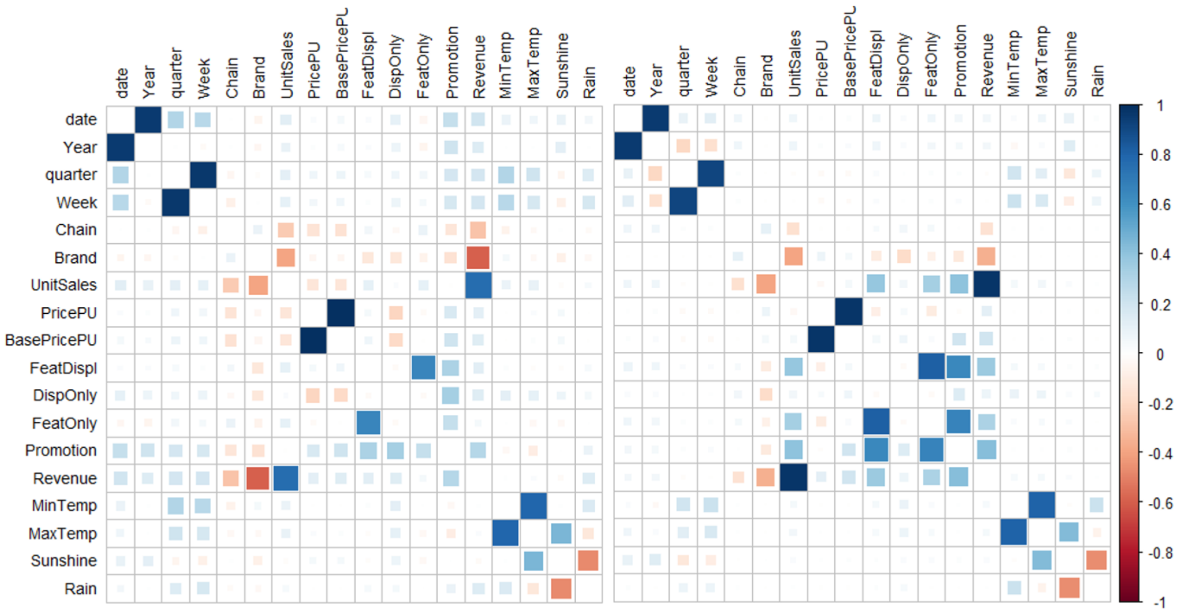

Figure 14 **Correlations among artificial data (left), correlations among real data (right).**

6.2. The predictive validity is equal between estimations based on a sample from p_G and p_{data}

In this section, we investigate the predictive validity of the artificial and real data. We employ a logistic regression, neural network, random forest, decision tree and bagging and boosting to investigate the . We evaluate the performance in terms of hitrate, TDL and Gini coefficient (Breiman 2001, Lemmens and Croux 2006, Risselada et al. 2010).

First, we estimate an artificial estimation on a sample from p_G (n = 2,816) and a sample from p_{data} (n = 2,499). Subsequently, we estimate a real estimation on 75 percent from the real sample and evaluate the predictive accuracy in terms of the hitrate, top-decile lift (TDL) and Gini coefficient on 25 percent of the real sample (n = 834) (Leeflang et al. 2017). We estimate the artificial estimation on 100 percent of the sample from p_G and evaluated on the same 25 percent from the real sample (n = 834).

We simulate both estimations a 1,000 times to create a distribution of hitrates, TDLs and Gini coefficients. Each time we train the estimations, the train and test set consist of a different random sample from the data. In this way, we account for the randomization in the selection of observations for the train or test set and are able to test for significant differences. To test whether the hitrates, TDLs and Gini coefficients are significantly different a Wilcoxon rank-sum test is employed. Where both variables are normally distributed, determined by a Shapiro-Wilk Normality test, a Student t-test is employed. All subsequent comparisons of predictive validity measures are tested for significant differences in a similar fashion.

6.2.1. Public churn data set. The results from the simulations show that the differences in hitrate are in some cases not significant or the artificial data outperforms the real data. In case of a neural network, both estimations resulted in an hitrate of 85.55 percent (W = .99, p = .93; t = -.46, p = .64). Surprisingly, the performance of a logistic regression estimated on artificial data significantly outperforms the model estimated on the real data (86.53 percent vs. 86.05 percent). In terms of top decile lift, the artificial data outperforms the real data in case of a logit (4.20 vs. 3.50). In the case of a neural network, the TDL in the real and artificial estimation is 3.58 and 3.43. Contrary to believe until now, artificial data are not only equal in performance, it is even outperforming estimations on real data. Figure 15 summarizes the results from our experiments.

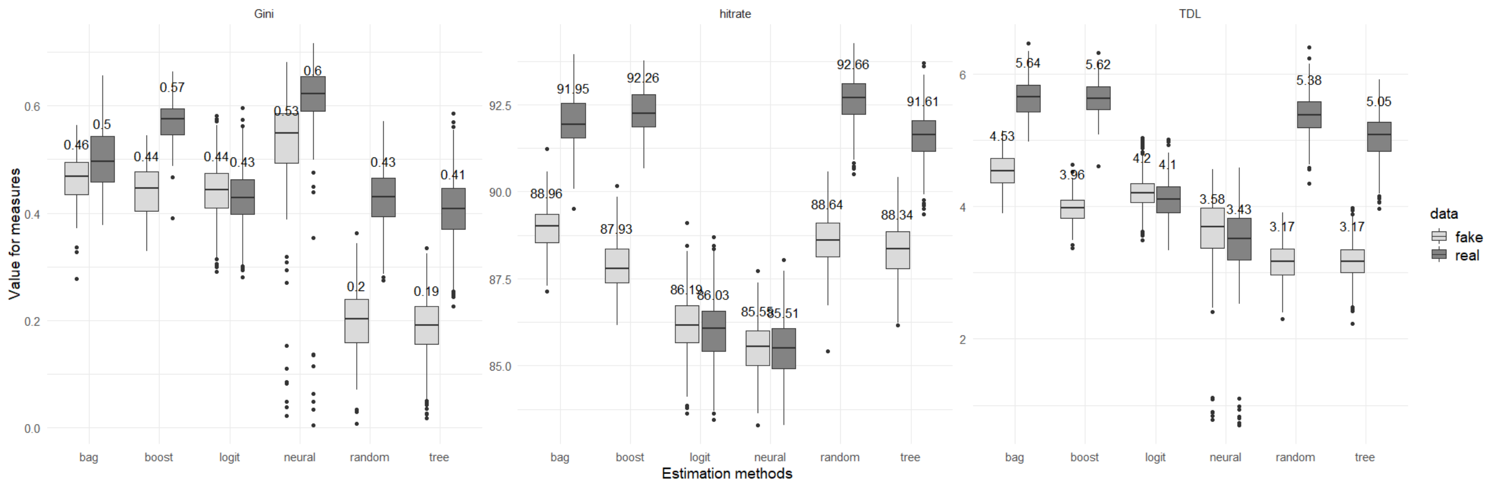

Figure 15 **The hitrate, TDL and Gini coefficient for our estimation methods in case of the public churn data set.**

6.2.2. Real churn data set. In case of the real churn data set, the predictive accuracy of the real and artificial churn data set is assessed by the same machine learning methods (see Figure 16). Here, bagging and boosting are excluded, since it is computationally too expensive to estimate the models on such a large data set (n = 1,262,423).

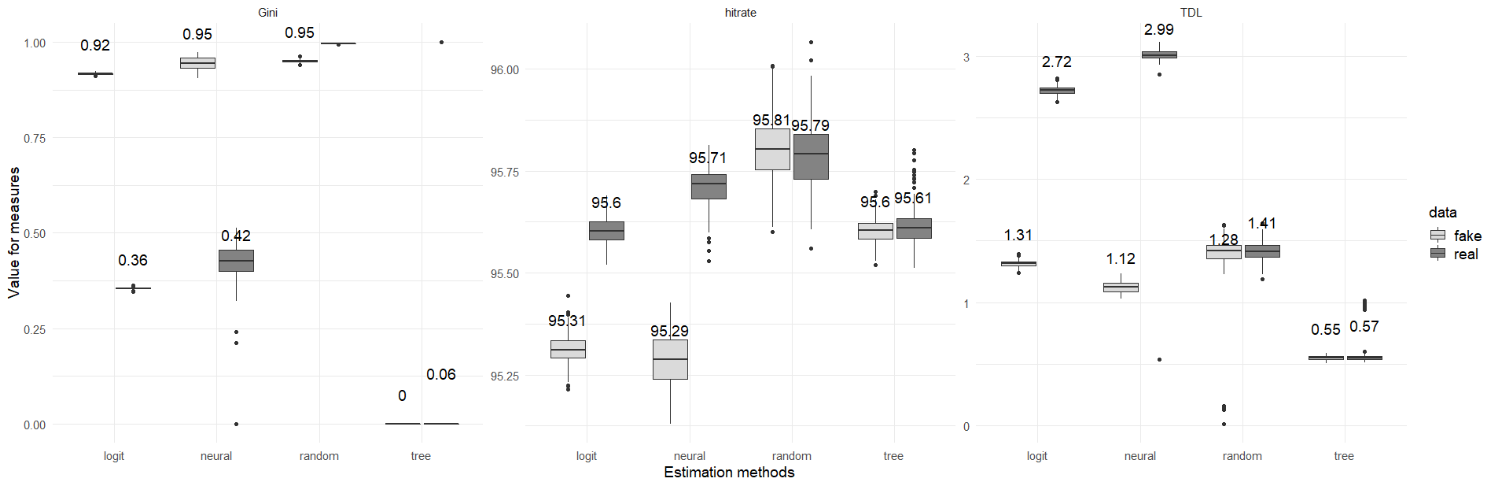

Figure 16 The hitrate, TDL and Gini coefficient for our estimation methods in case of the real churn data set.

In Figure 16, we observe similar results. For a random forest, the hitrate of an artificial estimation is outperforming the hitrate of a real estimation (95.81 percent vs. 95.79 percent). When we investigate the Gini coefficient, the logit and neural network estimated on artificial data outperform the real estimations. For the TDL, the performances of the random forest and decision tree are very close.

6.2.3. Lemonade sales data set. For the lemonade sales data set, we evaluate the predictive validity of lemonade sales from a SCAN*PRO estimation. A predictive validity measure that is dimensionless across models is the Mean Absolute Prediction Error (MAPE) (Leeflang et al. 2015, Schneider et al. 2018). The MAPE allows for a comparison of the predictive validity of two estimations regardless of the scale of the dependent variables. Additionally, we compare our estimations to a naive model that takes the value of the dependent variable in $t-1$ as a prediction for the next period $(t+1)$. These measures are the Relative Absolute Error (RAE) and Theil's U-statistic. When the outcomes of these statistics are less than one, the model outperforms a naive model (Leeflang et al. 2015). This scenario would imply that estimations on artificial data are worthwhile.

To compare the predictive validity of the artificial estimations, we exclude the last 71 observations of the real observations from the estimation data of the real model. We use rest of the observations to estimate the model. For completeness, we assess the predictive validity of the real model in terms of the APE, ASPE, RASPE, MAPE, RAE and Theil U-statistic (Leeflang et al. 2015). We assess the predictive validity of the artificial estimations on the same last 71 observations from the real observations and the same measures. We present the result of the assessments in Figure 17. While evaluating these measures, one has to take into account that the GAN is trained on 169 observations, which is considered to be extremely little information for the application of GANs.

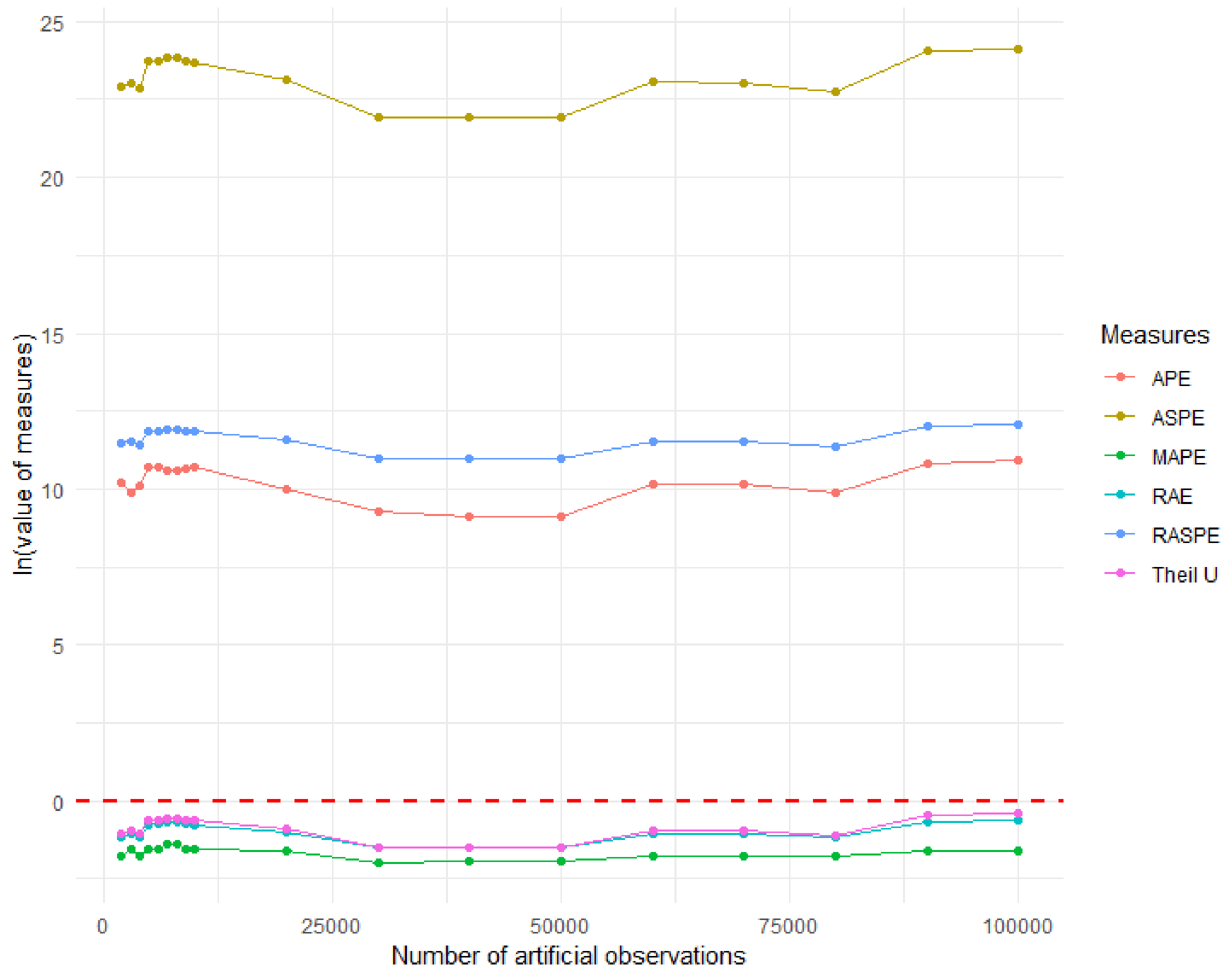

Figure 17 **The log-measures of APE, ASPE, MAPE, RASPE, RAE and Theil U statistic for the number of artificial observations. The red line indicates a naive model that takes $t-1$ as a prediction for t. In general, we can interpret the measures as the lower the better.**

In Figure 17, we present the results from eighteen artificial estimations that differ in the number of artificial observations. In this way, we investigate how the predictive validity differs for a variety of sample sizes from the GAN. It is interesting to note that the the predictive accuracy increases, as we increase the number of artificial observations. As n increases the predictive accuracy improves and starts to decrease when $n > 45,000$, which indicates that there is an ideal point around 45,000 - 50,000 observations (see Figure 17).

Subsequently, we experiment with the specification that results in the best predictive validity. We settle for an estimation on 45,000 artificial observation. Figure 18 shows how the artificial estimation on 45,000 observations is able to predict the real sales data quite accurately. The results indicate that the artificial estimation is not able to outperform the real model (MAPE = .071 vs .151). Nonetheless, the artificial estimation is able to predict the real sales with acceptable precision. For example, the artificial estimation outperforms a naive model (RAE = .207, Theil U = .194). Finally, the artificial estimation results in an APE of 3,148, ASPE of 2,649,830,326 and RASPE of 51,476.

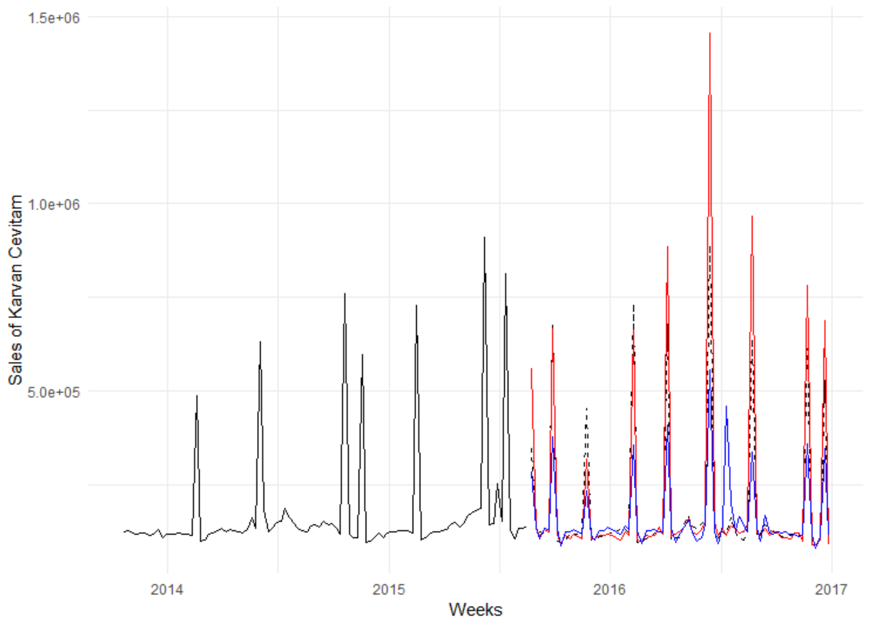

Figure 18 The predictive validity of the artificial estimation (blue) vs the real estimation (red) (n = 71).

6.3. Comparing the parameter estimates based on p_G and p_{data}

In this section, we investigate whether the artificial estimations are able to maintain the ability to derive prescriptive insights. Specifically, we compare the parameter estimates of churn models and a SCAN*PRO model for the lemonade data set.

6.3.1. Public churn data set. We use the public churn data set to estimate a logistic regression. We compare the parameter estimates from the artificial and real estimation. First of all, in both estimations, high VIF values arise in the variables: *VMailMessage, DayCharge, EveCharge, NightCharge* and *IntlCharge*. Therefore, these variables are excluded from both estimations. Finally, the parameters of the logit estimated on a sample from p_G and a sample from p_{data} correlate highly ($r = .99$, $p < .00$).

To test whether both models have similar parameters and covariance matrices, we employ a Hausman test. The Hausman test results in a significant difference between the real and artificial estimation ($\chi^2 = 194$, $p < .00$). This implies that the two estimations are significantly different from the real estimation, taking into account the parameters and variance. While being significantly different, the artificial estimation could still be considered useful. When we compare the estimations in Figure 19, they appear to be visually similar.

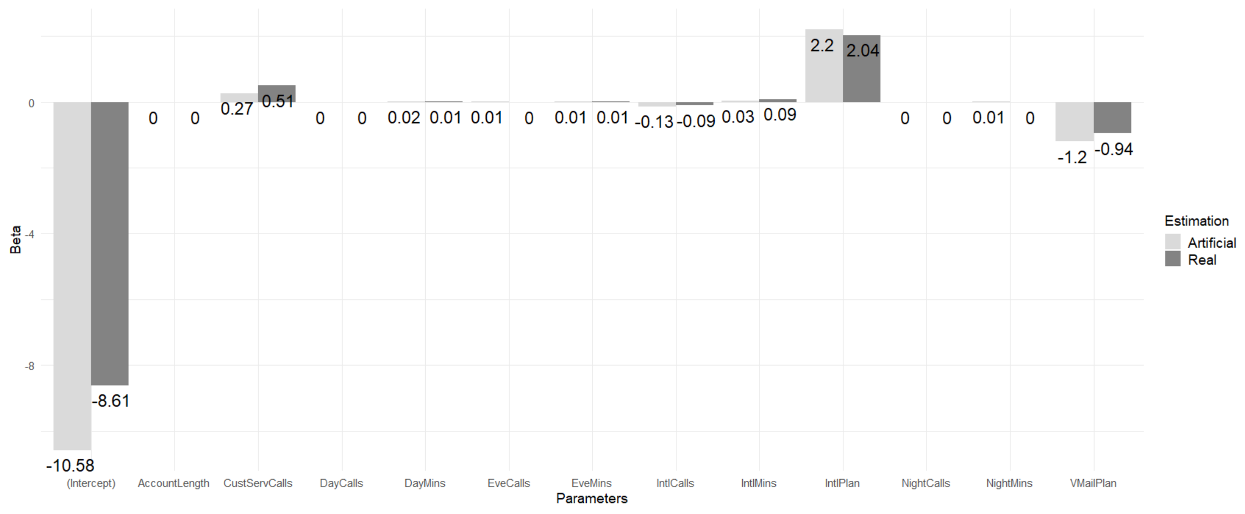

Figure 19 Parameter estimates of logistic regression in case of the public churn data set. We round the parameter estimates after the second digit behind the decimal point.

In the artificial estimation, the parameter estimates are of the same sign compared to the real estimation. Only the variable *AccountLength* has an opposite sign from the sign in the real estimation. However, *AccountLength* is not significantly different from zero in both estimations. Schneider et al. (2018) define the Mean Squared Error (MSE) to compare the parameter estimates as follows:

$$\text{MSE} = \frac{1}{J}\sum_{j=1}^{I}\left(\hat{\beta}_j - \beta_j\right)^2, \tag{28}$$

where J are the total number of parameters, β_j are the parameters estimates from a sample from p_{data} and $\hat{\beta}_j$ are the parameters from a sample from p_G. For our two estimations the MSE is only .024. This implies that on average our artificial parameters only deviate .024 from our real parameter estimates.

Subsequently, we apply a least absolute shrinkage and selection operator (LASSO) logistic regression on both the real and artificial sample. In other words, we include a L^1-norm penalty on the parameters to the loss function of the logistic regression. Similar to L^2-regularization, L^1-regularization pulls the parameter estimates strongest in the direction where the curvature of the loss function is the weakest (i.e. low eigenvalues of the Hessian matrix of the loss function with respect to the parameters). As a result, we automatically select the parameters that contribute most to the decrease of the loss function. We opt for L^1-norm parameter penalty, because the resulting parameters are more sparse compared to L^2-regularization. In Figure 20, the estimation is clearly more sparse compared to the logit estimation in Figure 19. The estimation results in a MSE of exactly zero (Schneider et al. 2018). The parameter estimates are exactly equal between the real and artificial estimation. In conclusion, we argue that the artificial data, in case of the public churn data set, is able to maintain the ability to derive marketing insights.

6.3.2. Real churn data set. In case of the real churn data set, we employ a logistic regression on a sample from p_{data} followed by an estimation on a sample from p_G. An estimation on a sample from p_{data} did not suffer from multicollinearity. All the VIF values from the estimation are below the critical value of 5.

In the first real estimation all variables have a linear specification. We transform the variables *size of policy*, *address size*, *social class*, *income* and *stage of life* from a linear to a dummy specification which decrease the BIC to 424,487, 424,457, 424,309, 424,298 and 419,432, respectively. Changing the specification of the variables *education*, *complaints* and *distance to store* does not lead to a decrease in BIC (419,432, 419,459 and 419,441). We test the continuous variables *age*, *relationship duration* and *declaration amount* for a quadratic specification instead of a linear specification. This leads to a decrease in the BIC to 418,479 and 418,209. We find that specifying *declaration amount* in a quadratic form

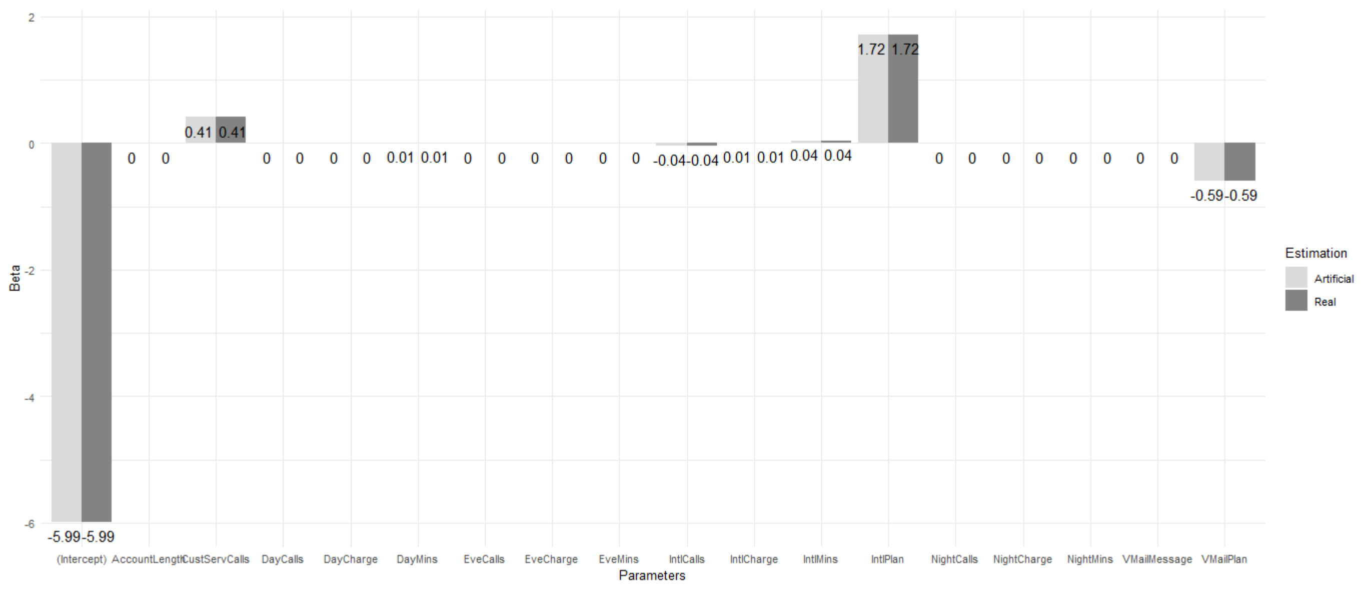

Figure 20 **Sparse estimations with L^1-regularization in case of the public churn data set.**

does not decrease the BIC (BIC = 418,224). Therefore, we prefer a linear specification of this variable. Finally, the variable *BSR.groen* has the highest p-value and removing the variable leads to a decrease in BIC to 418,196. Finally, we obtain a model with only significant variables and significantly outperformed a null-model ($\chi^2 = 36{,}387$, $p < .00$).

Subsequently, we use the same variables to estimate an artificial estimation. Surprisingly, this leads to an estimation that suffers from a missing level in the dummy variable *stage of life* in the artificial data. Removing this level from the variable, leads to a substantial decrease in the BIC in the *real* estimation to 408,865. From Figure 21 and 22, we observe that the coefficients from both estimations are moderately correlated ($r = .43$, $p < .00$). The weaker correlation is also reflected in the MSE of the parameter estimates (MSE = 28.35). Subsequently, the Hausman test shows a significant difference between the two estimations ($\chi^2 = 4{,}974$, $p < .00$).

In a similar way to the public churn data set, we apply L^1-regularization to the logistic regression in an attempt to obtain a sparse estimation. As a result, we obtain a MSE of .082 between the artificial and real parameter estimates in Figure 23. This implies that the GAN is able to generate the most important variables. Therefore, we argue that the artificial estimation is able to retain the prescriptive value of the real estimation.

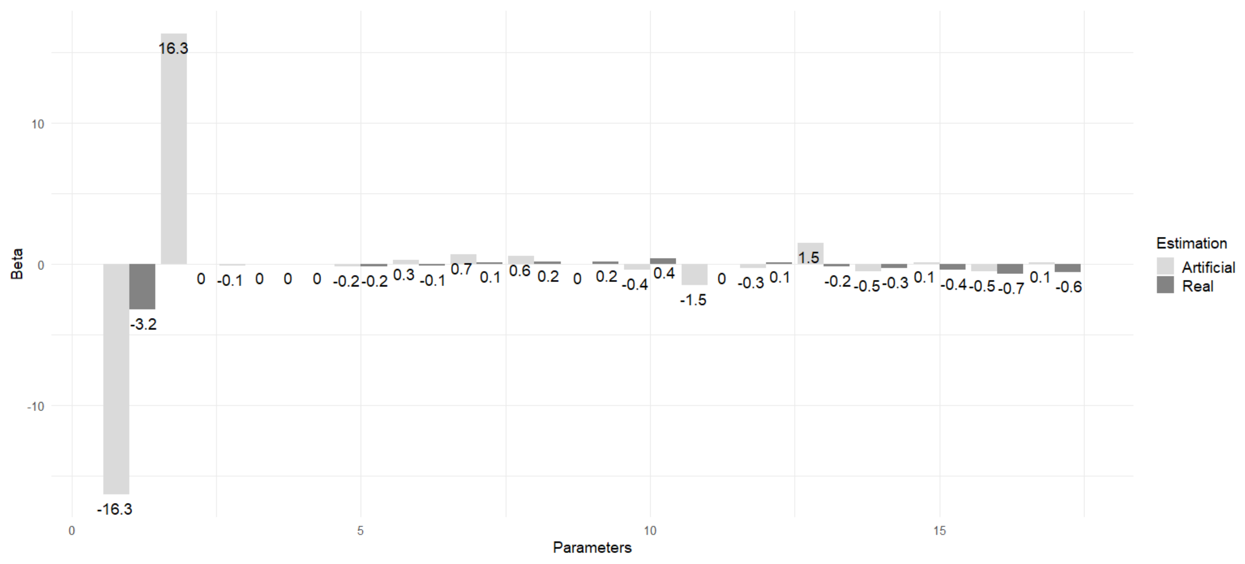

Figure 21 **Parameter estimates of the real and artificial estimation for the real churn data set (1/2).**

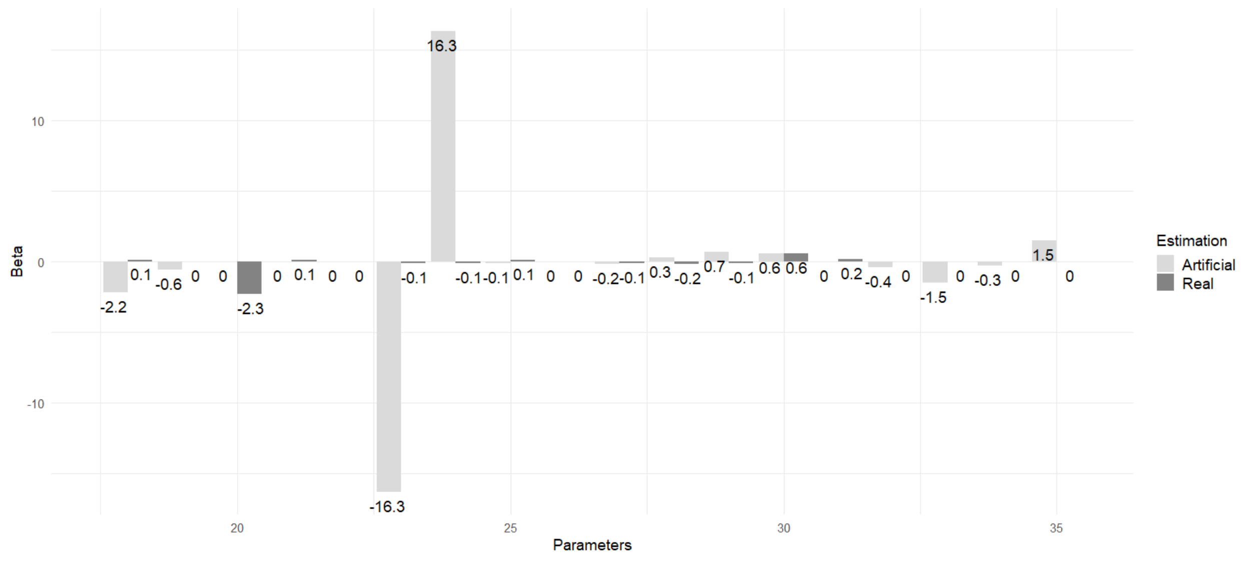

Figure 22 **Parameter estimates of the real and artificial estimation for the real churn data set (2/2).**

6.3.3. Lemonade sales data set. In case of the lemonade sales data set, we estimate a SCAN*PRO model on a sample from p_{data} (Leeflang et al. 2015, Schneider et al. 2018). The SCAN*PRO model has enabled already over 3,000 practitioners to get insight in a multiplicative effect between sales units and price, competitive price, advertising, in-store display and feature advertising (Schneider et al. 2018). In our estimation, we use weekly

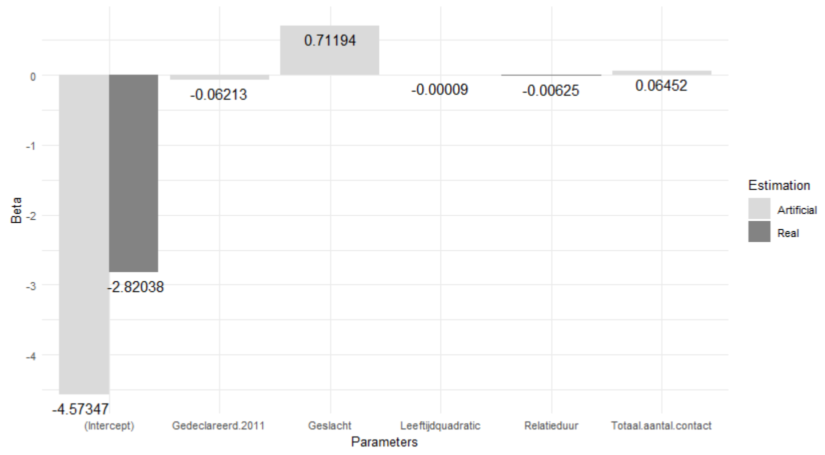

Figure 23 **Sparse estimations with L^1-regularization in case of the real churn data set. We round the parameter estimates after the fifth digit behind the decimal point.**

lemonade sales data from a supermarket chain in the Netherlands. We include additional weather information, account for the discontinuation of a specific supermarket and include Google Trends data on the release of an additional product of the lemonade brand. The SCAN*PRO specification is defined as follows (Leeflang et al. 2015, Schneider et al. 2018):

$$S_{ijt} = \alpha_{ij} \prod_{j=1,\ldots,3}^{J} [P_{ijt}^{\beta_{1ij}} \prod_{l=1,\ldots,4}^{L} (\beta_{lij}^{D_{lijt}})] T_t^{\beta_{16}} \beta_{17}^{R_t} \beta_{18}^{Sun_t} \beta_{19}^{D_t} e^{\varepsilon_{ijt}}. \tag{29}$$

Here, S_{ijt} represents the unit of sales at store i, for brand j and week t, P_{ijt} is the price which results in a price elasticity. Furthermore, D_{ijt} represent the promotional variables such as feature, feature and display, display and price promotion, T_t is the temperature in Kelvin, R_t is the duration of rain in .1 hour, Sun_t is the duration of sunshine in .1 hour and D^t is a dummy variable for the discontinuation of one of the supermarket chains.

Subsequently, we adapt the SCAN*PRO specification to one supermarket and one brand of lemonade (i.e., cross-sectional). We have to opt for a cross sectional model, because the DCGAN was not able to recreate panel data (see section 5.3.3). Finally, we specify the following estimation with taking the log in Equation 29:

$$\ln S_t = \ln \alpha + \sum_{j=1,\ldots,3}^{J} [\beta_j \ln P_{jt} + \sum_{l=1,\ldots,4}^{L} (\ln \beta_{lj}) D_{ljt}] + \beta_{16} \ln T_t + \ln \beta_{17} R_t + \ln \beta_{18} Sun_t + \ln \beta_{19} D_t + \varepsilon_t \tag{30}$$

For the estimation on the real sample, we perform a test for multicollinearity, autocorrelation, heteroskedasticity and normality (Leeflang et al. 2015). First of all, we account for multicollinearity in line with (Leeflang et al. 2015, p. 185). Second, we do not encounter first-order autocorrelation ($\rho = .13$, $p = .09$) or heteroskedasticity with a Breusch-Pragan test ($\chi^2 = 27.47$, $p = .16$) nor non-normality of the residuals ($\chi^2 = 1.23$, $p = .51$).

Subsequently, we generate an estimation on an artificial sample from p_G. We compare both estimations in terms of the parameters of the estimations with a Pearson's correlation test, MSE and an Hausman test (Schneider et al. 2018). We observe that the parameters from the artificial estimation correlate highly with the real parameters ($r = .95$, $p < .00$). However, the Hausman test shows that there is a significant difference between the real and artificial estimation ($\chi^2 = 10$, $p = .01$). Although the estimations are significantly different, one could argue that the prescriptive function of the model is contained, as most of the parameters are similar (see Figure 24). This is also reflected in the low value of the Hausman test statistic. For example, the own price elasticity parameter is -1.542 in the artificial estimation versus -1.809 in the real estimation. Finally, we observe a MSE of 1.12, which indicates that the real parameter estimates deviate 1.12 on average from the artificial parameter estimates (see Figure 24).

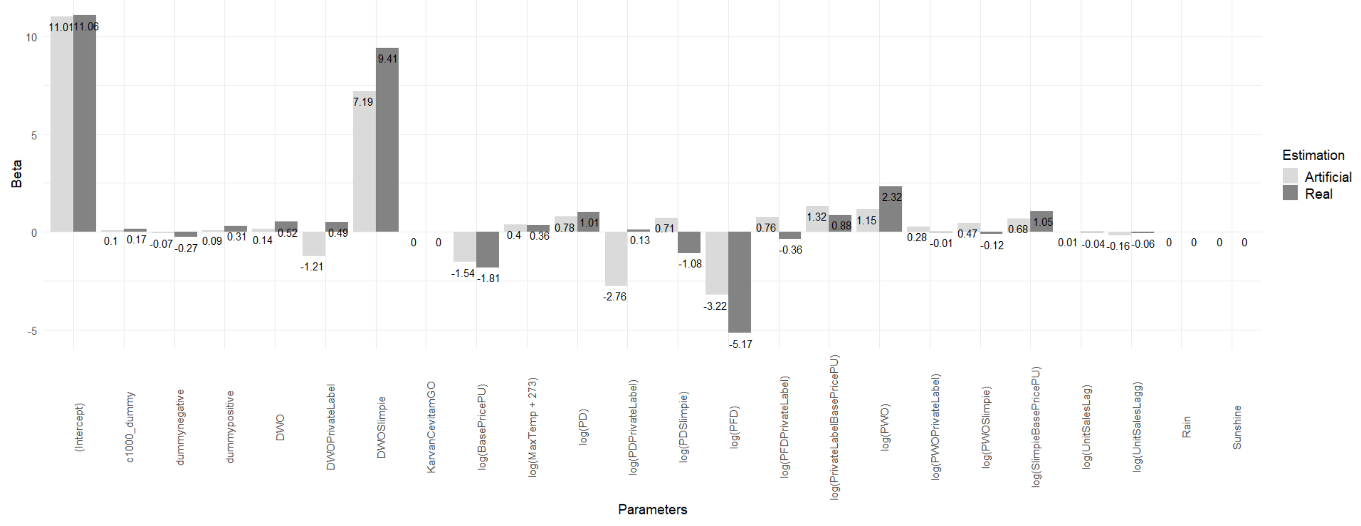

Figure 24 **Parameter estimates of the real and artificial estimation for the SCAN*PRO estimation.**

Surprisingly, the artificial estimation did not suffer from autocorrelation, resembling the real estimation (ρ = -.003, p = .70). Compared to the real estimation, the artificial estimation does suffer from heteroskedasticity (χ^2 = 1,104, $p < .00$). Next, we conduct a Jarque-Bera test on all the residuals and a Shapiro-Wilk test for normality on a random sample of 5,000 residuals from the artificial estimation (Leeflang et al. 2015). Compared to the real estimation, we can not assume that the residuals are coming from a normal distribution (χ^2 = 11,113,000, $p < .00$; W = .95, $p < .00$). Leeflang et al. (2015) note that with increasing samples sizes, the null-hypothesis of normally distributed errors is rejected more often, because even minor deviations from normality signify a violation of the assumption. To investigate the consequences of non-normal distributed errors, we perform bootstrapping. Bootstrapping functions as a remedy for non-normality issues, but also functions as verification of the initial model. When we apply bootstrapping we observe that the price combined with a feature and display for the brand (p = .39) and *rain* (p = .23) arise as insignificant variables.

In contrast to the real estimation, the artificial estimation suffered from multicollinearity. The VIF values for two of the explanatory variables are higher than 5. Therefore, we generate more data to see whether additional artificial data leads to more variation in both variables and to reduce the correlation between both variables (n = 100,000). This procedure did not prevent the estimation from suffering from multicollinearity (VIF = 8 and 10), neither did it improve the fit of the estimation (R^2 = .89, $p < .00$) nor resemble the real estimation better (χ^2 = 7,440, $p < .00$). A second procedure is to delete one of both variables from both the real and artificial estimation, next to omitted variables bias, deleting the variable *PrivateLableBasePricePU* (χ^2 = 13,906, $p < .00$) or *BasePrice* (χ^2 = 7,590, $p < .00$) did not lead to a closer resemblance between the real and artificial estimation.

Finally, we apply L^1-regularization to obtain sparse parameter estimates. In Figure 25, we display the sparse parameter estimates. The MSE decreases to .98 from 1.12 in the estimation *without* L^1-regularization in Figure 24. In conclusion, we argue that the artificial estimation is able to retain the prescriptive value of the real estimation.

6.4. Convergence

In this section, we compare the different GAN architectures in terms of performance to approximate the real data distribution p_{data}. As we described, theoretically GANs approximate the real data distribution. However, we also described that in practice there are

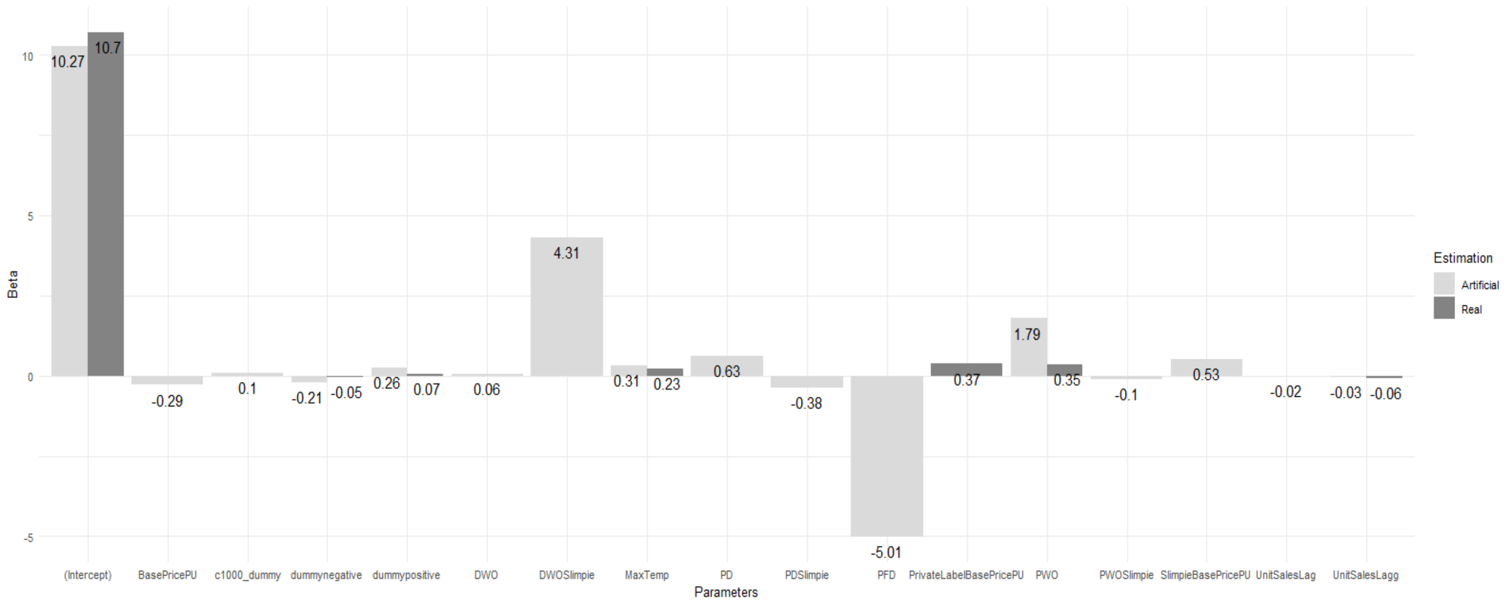

Figure 25 **Parameter estimates of the real and artificial estimation for the SCAN*PRO estimation with L^1-regularization.**

many roadblocks in reaching this state of convergence (see section 3.4). Therefore, the literature identified Recurrent GANs, DCGANs, WGANs, WGAN-GPs as improvements to overcome the difficulty of training GANs (Radford et al. 2015, Salimans et al. 2016, Arjovsky et al. 2017, Gulrajani et al. 2017).

We evaluate the architectures over the three data sets in terms of accuracy of the discriminator over the artificial samples, the correlation between a sample from p_{data} and p_G, loss function values and univariate distributions of the artificial samples. The architecture for the DCGAN, WGAN and WGAN-GP we adapt is visible is available in Figure 8. For the RGAN, we replace the generator with a LSTM layer.

6.4.1. Accuracy of the discriminator. For the three data sets, we first evaluate the accuracy of the discriminator on the artificial samples $G(\boldsymbol{z})$ over the iterations (see Figure 26). In other words, we plot $D(G(\boldsymbol{z}))$ over iterations. We separate the comparison in terms of accuracy of the discriminator between Wasserstein loss based GANs and other GANs, because the discriminator acts as a critic rather than a classifier. Therefore, we are unable to obtain an accuracy for the Wasserstein GANs, because the activation function of the discriminator is linear instead of a sigmoid.

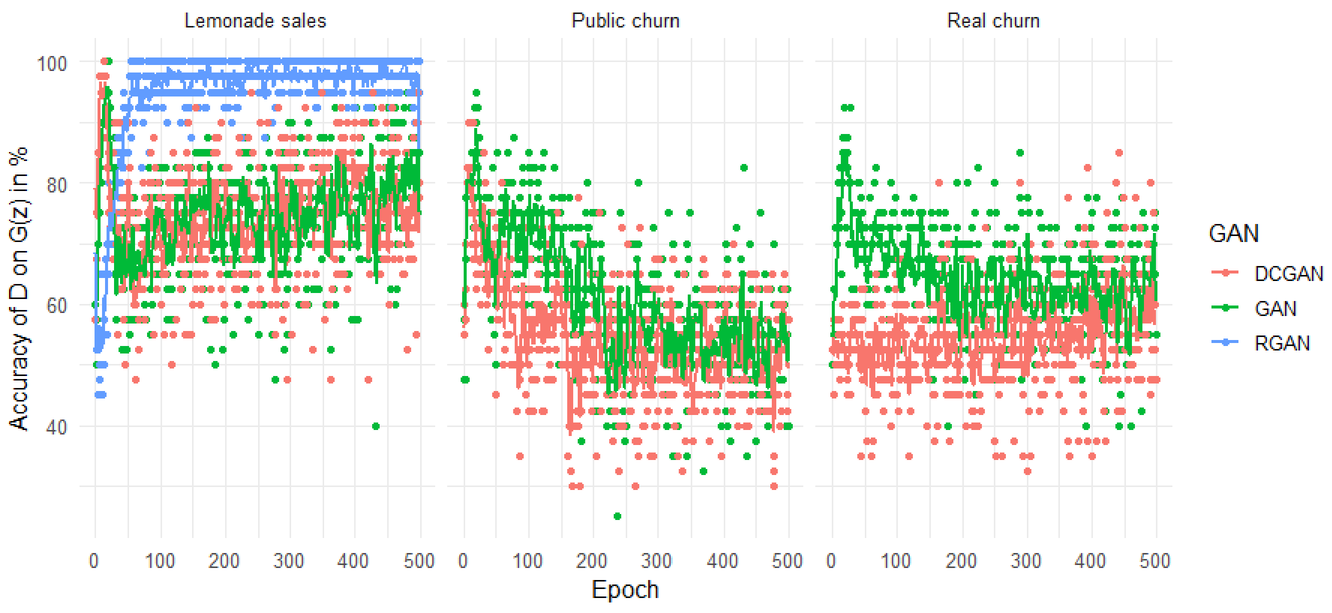

Figure 26 **Comparison of the accuracy of the discriminator over iterations. One epoch in the plot represents 100 iterations of training. The line is a running average with a width of 5. For this experiment we did not use label smoothing, because then we would be unable to calculate the percentage of correctly classified artificial observations.**

From Figure 26, we observe that for all data sets the GAN and DCGAN are uncertain whether the artificial samples are from p_{data} or p_G. However, for the lemonade sales data set that the accuracy of the discriminator in the RGAN goes to 100 percent rapidly over the artificial observations. On average the discriminator predicts the artificial samples to be artificial 94.59 percent correctly. Indicating that the generator is unable to improve due to the lack of gradient from the discriminator. We make several attempts to increase and decrease the capacity of the discriminator and generator in the RGAN. Unfortunately, this did not lead to improve the accuracy of the discriminator over the artificial samples.

The GAN and DCGAN show for all the data sets that the discriminator is more uncertain about whether the artificial samples are from p_{data} or p_G. For the lemonade sales data set, the average accuracy of the discriminators for the GAN and DCGAN over $G(\boldsymbol{z})$ is around 74.6 percent in both cases. For the public churn data set that the average hitrate of the discriminators is 60.92 percent and 54.77 percent. The accuracy of the discriminators of the real churn data set is on average 64.44 percent for the GAN and 54.62 percent for the DCGAN. Theoretically, we expect that the situation where the accuracy of the discriminator is around 50 percent, the generator generates the highest quality of samples. Therefore, in the next section, we shift our attention to the ability of the generator to generate realistic samples over the iterations.

6.4.2. Correlation between samples from p_{data} and p_G. We investigate the correlation of the samples from p_{data} and p_G for the GAN architectures. For each 100 iterations, we take a sample from p_G and p_{data} to calculate the correlations. Specifically, for each sample, we obtain the relationships among the variables in a correlation matrix. We flatten the matrix into a vector and calculate the Pearson's correlation between the relationships of the variables in the two samples. In this way, we compare the relationships among variables in both samples. We present the results in Figure 27.

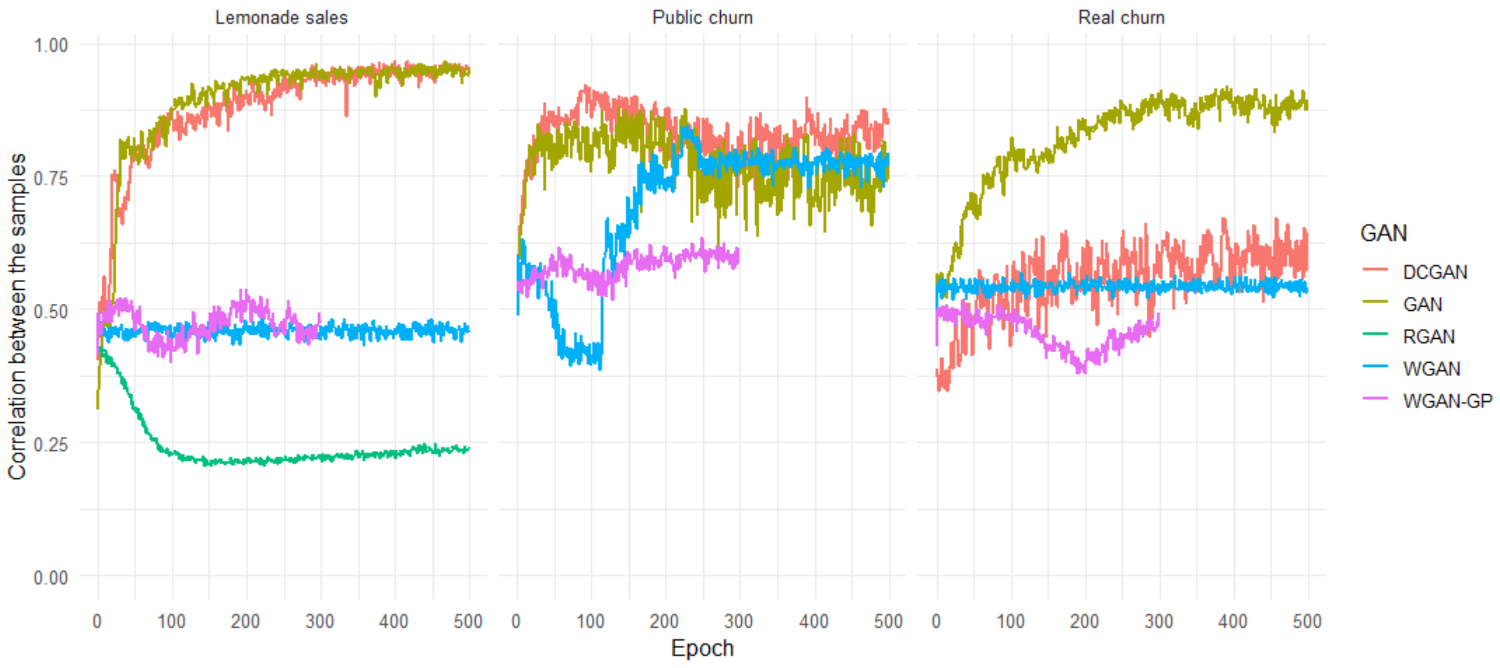

Figure 27 **Comparison of the correlation between samples from p_{data} and p_G over iterations. One epoch in the plot represents 100 iterations of training. We train the WGAN-GPs only 30,000 iterations due to the large computational cost of penalizing the gradient.**

For the lemonade sales data set, we observe that the generators of the GAN and DCGAN generate samples of high-quality early in training. Interestingly, this corresponds with our earlier observation that the discriminator is highly uncertain in the GAN and DCGAN. Unfortunately, the high certainty of the discriminator in the RGAN results in a decrease in the correlation between samples. In Figure 26, the discriminator becomes highly certain around 10,000 iterations. Exactly at the same iteration, the generator does not have a gradient to improve the quality of the artificial samples (see Figure 27). Similarly, the generators in the Wasserstein GANs perform well in early training, but the correlations do increase as training develops. In conclusion, the GAN or DCGAN perform best. However, in terms of computational expensiveness, the DCGAN is more efficient (see section 3.5.4). Therefore, we prefer the DCGAN for the lemonade sale data set.

For the public churn data set, we observe that the uncertain discriminators for the GAN and DCGAN lead to artificial samples of high-quality after only 10,000 iterations. In Figure 26, the accuracy of the discriminators in the GAN and DCGAN is highly uncertain. However, one could argue that the discriminators are almost *too* uncertain. As a result, in Figure 27, we observe that the correlation of the samples does not increase. This highlights the importance of a delicate balance between the two players.

In terms of the Wasserstein GANs, the WGAN samples first decrease in quality and after around 10,000 iterations the WGAN strongly improves the quality of the samples to around a Pearson's correlation of .75. In conclusion, there is a small difference between the correlation of the samples for the architectures. Ultimately, the DCGAN returns the highest correlation between the samples from p_{data} and p_G. Therefore, we prefer the DCGAN for the public churn data set.

Similarly, for the real churn data set, we find that the uncertainty of the discriminators result in a high correlation between the samples for the GAN. Surprisingly, the GAN outperforms the DCGAN, WGAN and WGAN-GP. Possibly, due to a large number of examples ($n = 1{,}262{,}324$), the data generating process is very complex to model. Therefore, we take advantage of the fully dense connected layers in the GAN compared to the DCGAN where the connections are sparse (see section 3.5.4).

6.4.3. Univariate distributions of the artificial samples. Additionally, we store the distributions of the first eight variables from artificial samples as images. We visualize the distribution of the artificial samples for all architectures and data sets over iterations here: `https://www.youtube.com/channel/UCNA5DwzV4ii-6NC-4epRHUA/videos`. We present the original univariate distribution of the first eight variables of the data sets in Figure 28.

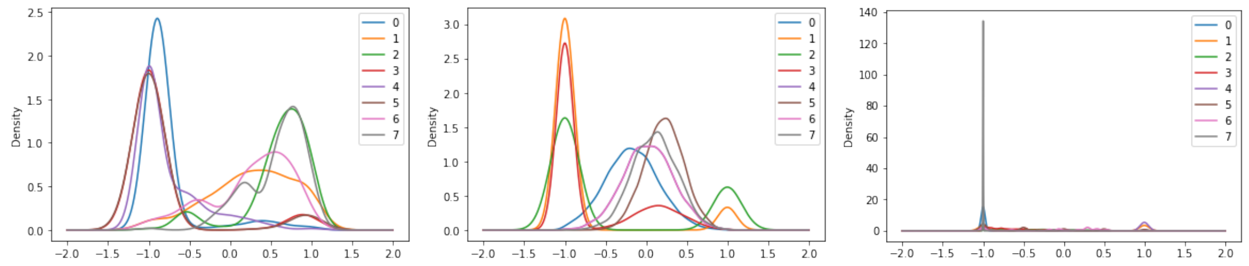

Figure 28 **The univariate distribution of the first eight variables from the respective data sets in the following order: the lemonade sales data set, public churn data set and real churn data set. For clarity, we create playlists for each data set which show the development of training for all the architectures. For the lemonade sales data set:** `https://www.youtube.com/playlist?list=PL_hzHTwLKJxKn9qb-OOvbF2u4cvyxgivi`**, for the public churn data set:** `https://www.youtube.com/playlist?list=PL_hzHTwLKJxIqrtcl5sUltUVT3eBjF1NW`**, and for the real churn data set** `https://www.youtube.com/playlist?list=PL_hzHTwLKJxKzpEClcNsREOd2Ya5usNhz`**.**

In line with previous observations, we observe from the videos that the generators are able to represent the distributions early in training. Overall, around 10,000 iterations the distributions are represented by the generator. Interestingly, the distributions occasionally change shape completely, possibly due to the regularizing effect of the mini-batch sizes. Each time we sample a mini-batch the distribution of the variables is different. In conclusions, the videos of the distributions largely support our findings from earlier analyses (e.g., Figure 27 and 26). The GAN and DCGAN quickly converge to the original distributions from Figure 28. While the RGAN, WGAN and WGAN-GP experience more difficulty to converge to the original distributions.

6.4.4. Conclusion. We conclude with an interesting observation that the GANs do not require a high amount of iterations to result in a generator with high-quality samples. To illustrate, some of the most impressive results for the generation of celebrity face images are a result of training with 8 Nvidia Tesla V100 GPUs for 4 days (Karras et al. 2017). In this study, training a DCGAN takes around 1 hour on Google Colab which uses, depending on what is available as a free user, a single Nvidia K80, T4, P4 or P100 GPU.

Overall, the GANs and DCGANs quickly converge to represent the real distributions. This confirms the usage of DCGANs in our elaborated analysis in our results chapter 6. One explanation for the superior performance of DCGANs may arise from the statistical efficiency (see section 3.5.4). The number of observations in the public churn and lemonade sales data set is too low for the densely connected layers in a GAN. Therefore, we possibly benefit from the efficiency of the convolutional layers in the DCGAN.

Surprisingly, in case of the real churn data set, the GAN clearly outperforms the DCGAN in terms of correlation between the samples. A plausible explanation may be that the real churn data set consists of a large number of observations (n = 1,262,423). Therefore, we can profit from the densely connected layers in the GAN to approximate the relatively more complex data generating process.

The Wasserstein based GANs theoretically have attractive properties but do not outperform the GAN and DCGAN. One explanation may be that the distributions always overlap, because we scale the distributions p_G and p_{data} to lie between -1 and 1 (see section 5.1). Potentially, we do not take full advantage of the theoretical properties of Wasserstein GANs.

Finally, the RGAN performs the worst out of any of the architectures for the lemonade sales data set. One possible explanation may be that the introduction of an LSTM layer in the generator introduces a large number of weights with complex interdependencies (see section 3.5.5). The low number of observations in the lemonade sales data set only exacerbate the problem of having even more weights. To illustrate magnitude of this issue, consider that we have 169 observations and sometimes up to 6 million weights. As a result, the generator is not able to generate samples of any acceptable quality for the discriminator to become uncertain. In the following section, we perform robustness checks to the hyperparameters we use in this study.

6.5. Robustness checks

In this section, we perform robustness checks for the public churn data set. We aim to investigate the effect of different optimizers, activations, dropout, batch normalization and label smoothing. To start our experiment, we use the DCGAN defined with 50,000 iterations.

6.5.1. Optimizers. First, we only vary the optimizers for the generator and discriminator. From the literature, we select Leaky ReLU as activation function, but do not use dropout, a batch size of 10, a learning rate of .001, no batch normalization nor one sided-label smoothing.

Every optimizer except for the stochastic gradient descent (SDG) uses adaptive learning rates. The other optimizers scale the learning rate inverse proportional to the sum of squared gradients. The main differences between the adaptive learning rate optimizers are the way the previous gradients are remembered. The AdaGrad algorithm stores the sum of squared gradients, which can lead to a rapid decrease in the learning rate for some weights. This is also what we observe in Figure 29. Recall that the discriminator has the objective to maximize the log-likelihood and the generator the objective to minimize the log-likelihood (see Equation 1). After around 20,000 iterations the AdaGrad barely further maximizes the log-likelihood possibly due to possibly a low learning rate.

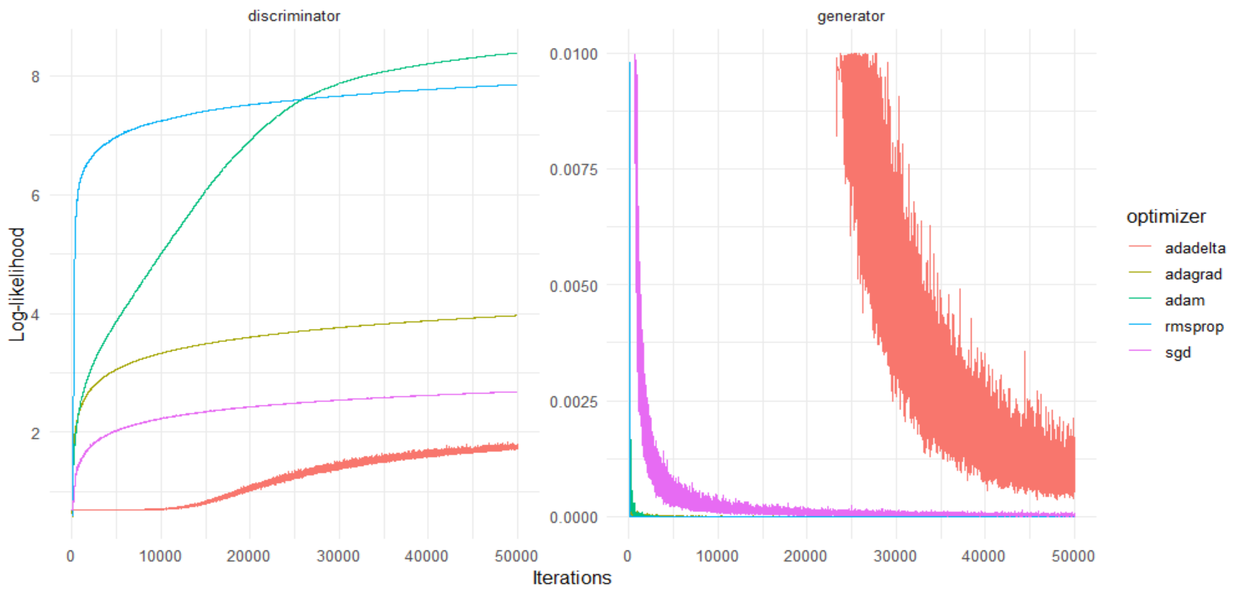

Figure 29 **Comparison of optimizers for the public churn GAN for 50,000 iterations. We cut the loss of the generator at .01 in the plot for a clear comparison.**

Due to the possible rapid decrease of the learning rate in AdaGrad, other optimizers take an exponentially decaying average of the gradients. This causes the most recent gradients to influence the adaptation of the learning rate the most. Kingma and Ba (2014) show empirically that Adam optimizer outperforms the other optimizers. From Figure 29, we derive a similar conclusion. In terms of the discriminator and generator, the Adam optimizer outperform the other optimizers. Therefore, in our robustness checks we proceed with the Adam optimizer. Only in our conclusion in section 6.5.8, we compare the results from the robustness checks with the architecture from our main results in chapter 6.

6.5.2. Activation functions. We vary the following activation functions for each layer in the discriminator and generator: tanh, sigmoid, ReLU, Leaky ReLU and softsign. The derivative of the sigmoid and tanh activation function saturates for large values of the input (e.g., $\sigma'(\boldsymbol{x}) = \sigma(\boldsymbol{x})(1-\sigma(\boldsymbol{x}))$ and $\tanh'(\boldsymbol{x}) = 1 - \tanh^2(\boldsymbol{x})$). However, the maximum of the derivative of the sigmoid function is .25. In the backward pass of backpropagation to update the weights, we multiply the derivative of the activation function with the gradient. If we use the sigmoid activation function in the hidden layers, we only pass 25 percent of the information to the weights. Therefore, the sigmoid activation function is discouraged as activation function in the hidden layers. The main motivation to use the ReLU activation function ($\max(0, \boldsymbol{z})$) is that the derivative is 1 for all positive inputs. In backpropagation, this enables the network to learn pass 100 percent of the information to the weight update. The downside is that the derivative is undefined for inputs that are zero. Therefore, Leaky ReLU returns a small constant α for the negative inputs to keep the unit active (i.e., $\max(0, \boldsymbol{z}) + \alpha \min(0, \boldsymbol{z})$).

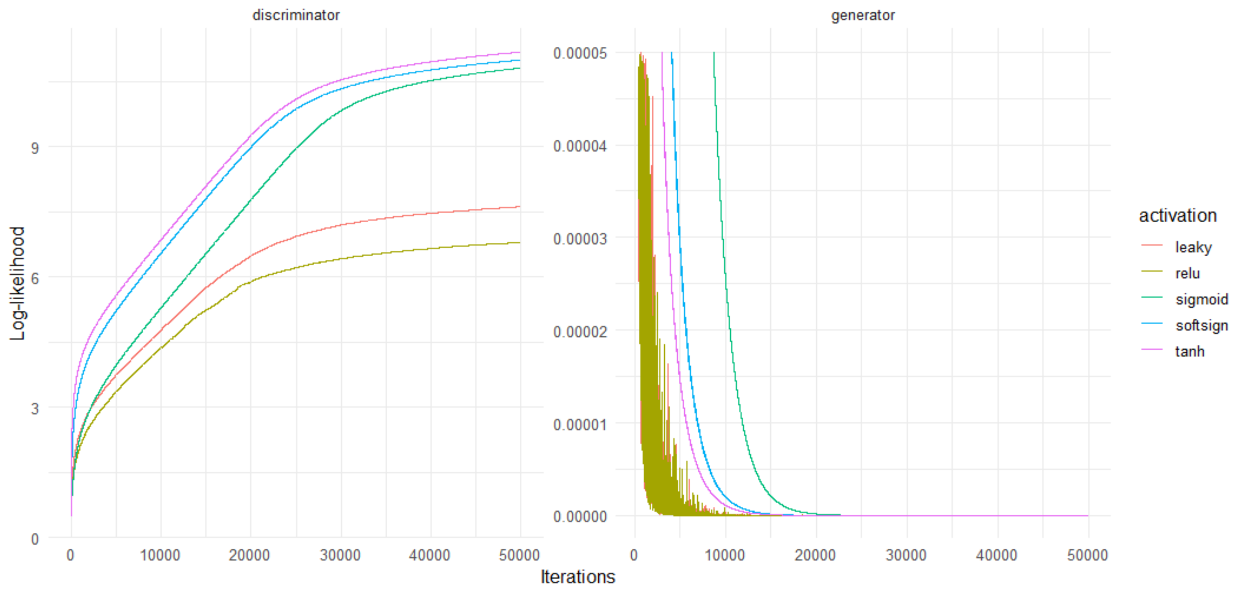

Figure 30 **Comparison of activations for the public churn GAN for 50,000 iterations. We cut off the value of the loss for the generator at .00005 and the iterations at 15,000 to get a clear comparison.**

From Figure 30, we observe that the Leaky ReLU and ReLU activation function perform best for the generator. For the discriminator, the tanh activation function performs best.

However, in case of the minimax game, the Leaky ReLU and ReLU result in a game where the generator is perhaps stronger which leads the discriminator to be less confident. For following robustness checks, we choose to use the Leaky ReLU in both the generator and discriminator.

6.5.3. Learning rate. The learning rates are often cited to be one of the most important hyperparameters. However, Kingma and Ba (2014) describe that the Adam optimizer is robust to the initialization of a random learning rate, because we adapt the learning rate during training. Figure 31 largely confirms their statement as the development of the loss for the learning rates is very close. Subsequently, we decide to continue with the learning rate .0001.

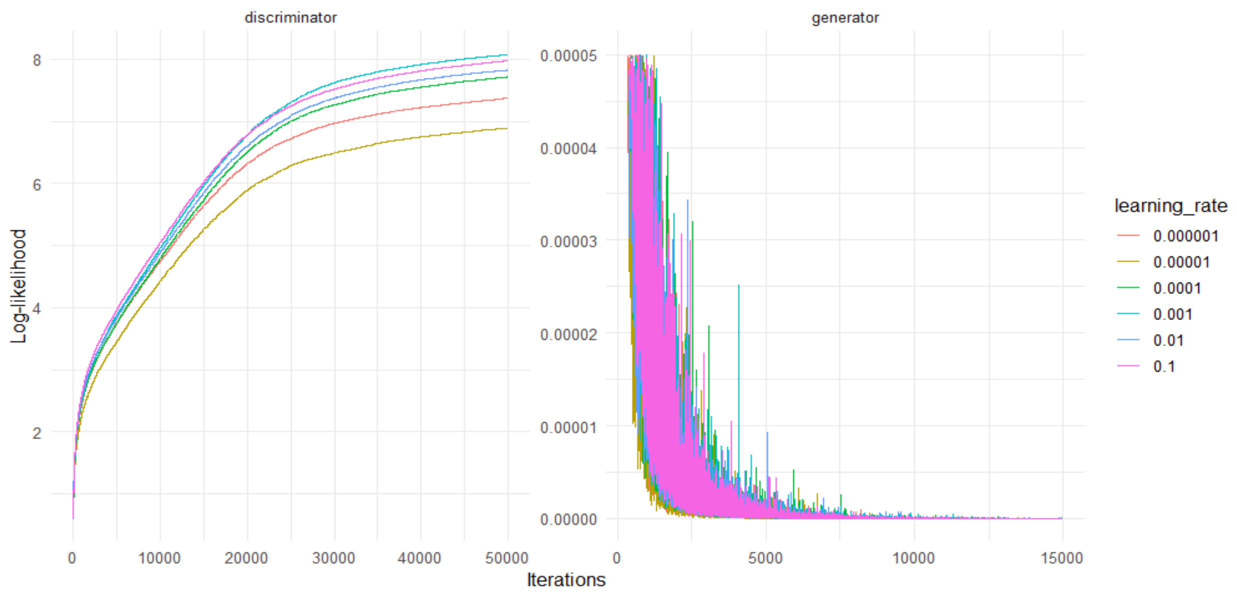

Figure 31 Comparison of learning rates for the public churn GAN for 50,000 iterations. We cut off the value of the loss for the generator at .00005 to get a clear comparison.

6.5.4. Dropout. For this robustness check, we only apply dropout in the architecture of the discriminator. Interestingly, the dropout in the discriminator also seems to induce randomness in the generator due to the minimax game. Figure 32 shows how with each 20 percent increase in the probability of dropout for the input and hidden units, the variance of the loss increases for both the generator and discriminator. A dropout of 40 percent seems to lead to a reasonable randomness in the loss of both players, while 60 percent seems to lead to too much randomness. Especially in the loss of the generator. This observation is in line with the original paper of Srivastava et al. (2014). The authors found that a dropout of 40 percent leads to the lowest generalization error. Therefore, we select to proceed this experiment with 40 percent dropout in the architecture.

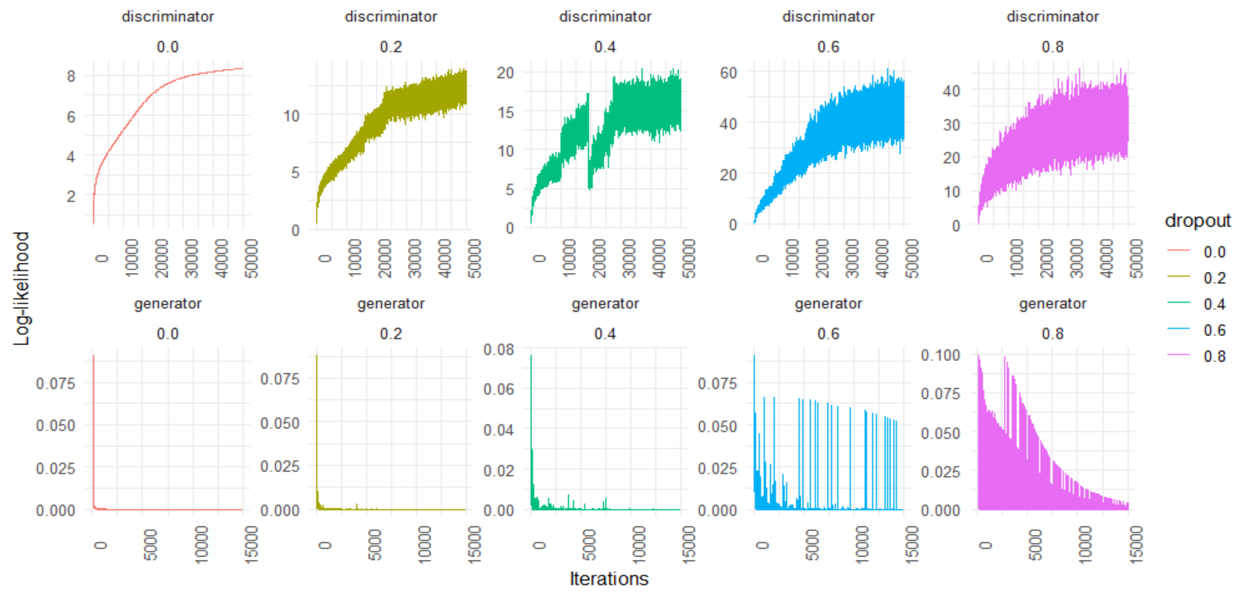

Figure 32 **Comparison of dropout for the public churn GAN for 50,000 iterations. We cut off the value of the loss for the generator at .1 to get a clear comparison.**

6.5.5. Batch size. In this section, we experiment with different sizes for the mini-batch size. As we described earlier, a small mini-batch size leads to a more noisy gradient. This has a regularizing effect and potentially allows us to escape local minima. A larger batch size allows us to compute more accurate gradients. From Figure 33, we observe that the smaller the batch size, the more variation in the loss function for both players. For subsequent experiments, we continue with a batch size of 250 observations.

6.5.6. Batch normalization. As we described in section 3.5.1, batch normalization allows all the layers in the network to learn at the same speed. From Figure 34, we observe that batch normalization has little effect on the optimization of both networks. One explanation could be that the networks are not sufficiently deep to have a large effect on the

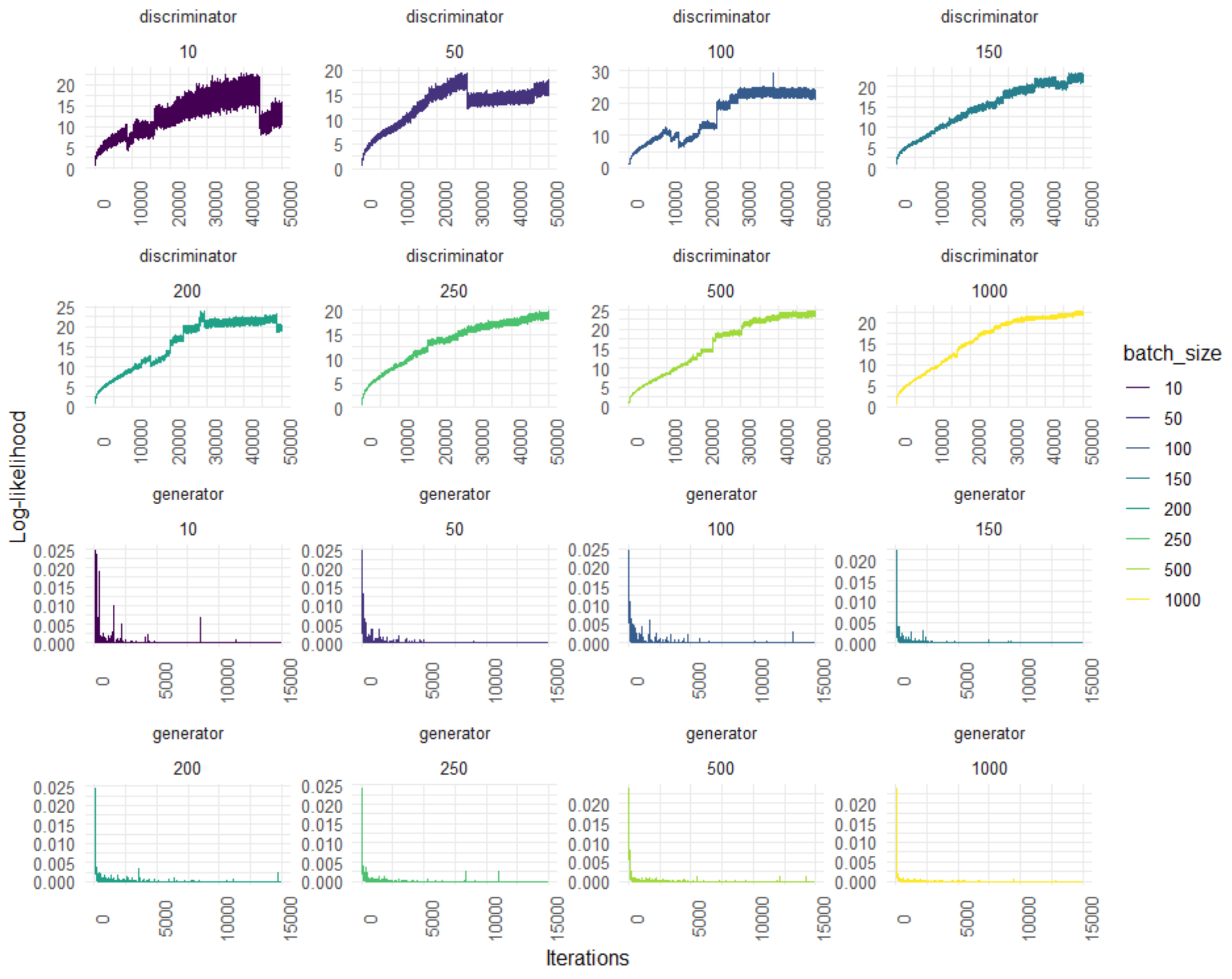

Figure 33 **Comparison of batch size for the public churn GAN for 50,000 iterations. We cut off the value of the loss for the generator at .00005 to get a clear comparison.**

optimization. Due to the theoretically attractive properties, we decide to include batch normalization for our final experiment.

6.5.7. Label smoothing. Finally, we test whether label smoothing reduces the certainty of the discriminator. From Figure 35, we observe that label smoothing indeed reduces the performance of the discriminator. Not only does it reduces the certainty of the discriminator in the early stages of training, it also does reduce the certainty late in the optimization procedure. Taking this evidence into account, we argue that it is useful to apply label smoothing.

6.5.8. Conclusion. The findings are largely in line with the AI literature and support our architectural choices in chapter 6 for the publicly available churn data set. Therefore,

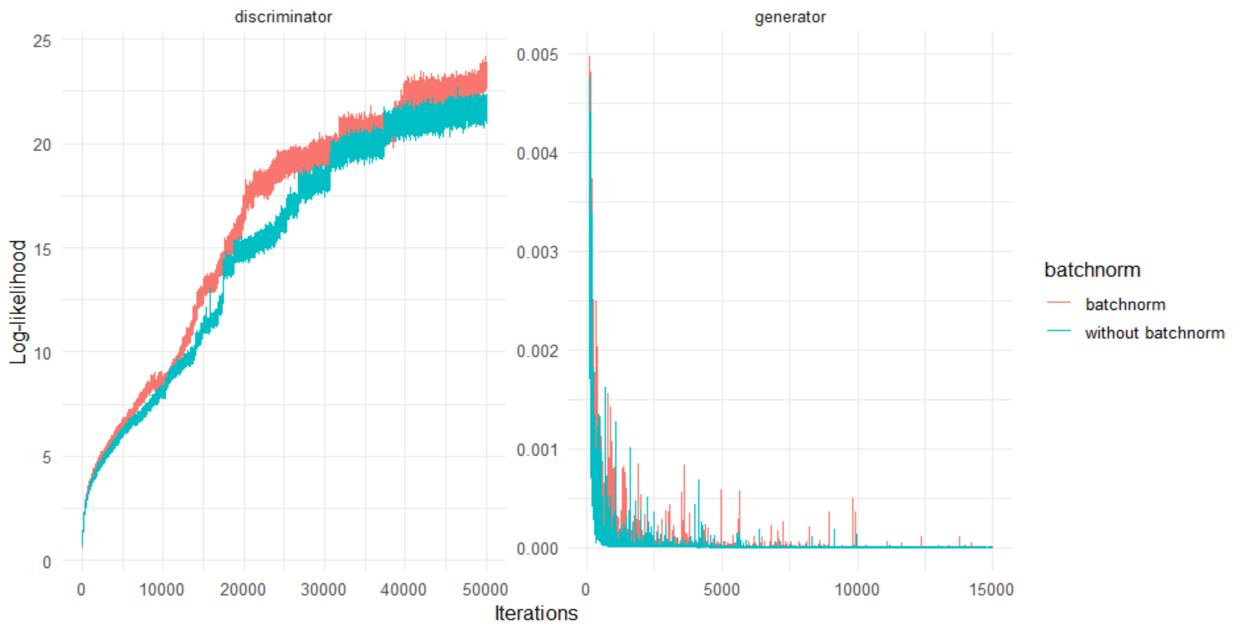

Figure 34 Comparison of batch normalization for the public churn GAN for 50,000 iterations. We cut off the value of the loss for the generator at .005 to get a clear comparison.

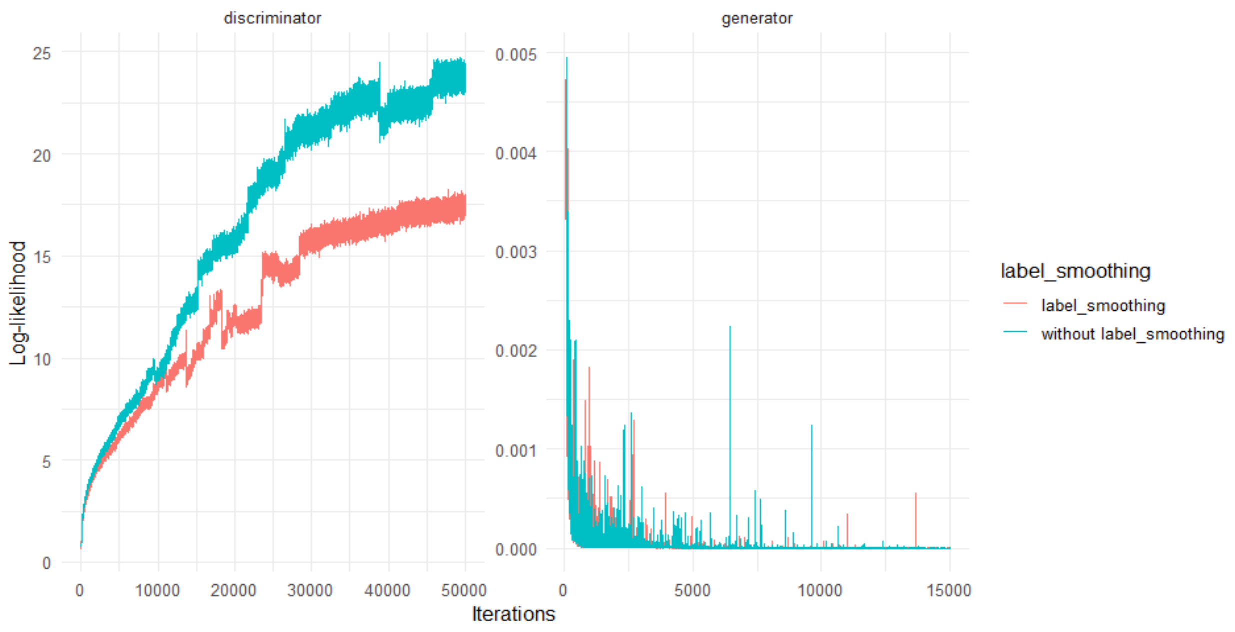

Figure 35 Comparison of label smoothing for the public churn GAN for 50,000 iterations. We cut off the value of the loss for the generator at .005 to get a clear comparison.

we motivate the use of the Adam optimizer, activation functions, a learning rate of .0001, dropout of 40 percent, batch normalization, a mini-batch size of around 250 observations and label smoothing in this study.

7. Conclusion, limitations and future research

In this chapter, we first conclude our paper. We also describe the limitations of our work and interesting directions for future research.

7.1. Conclusion

Until now, the marketing literature has not fully reaped benefits from the recent advancements in the AI literature. From the AI literature on generative modeling, we propose a variety of GANs to generate privacy-friendly artificial data. The methodology stands out compared to previous attempts in the ability to approximate the data generating process and the ability to share the artificial data for any purpose. As a result, practitioners and academics are able to share artificial data that helps to increase the predictive accuracy of estimations for practitioners or the ability to derive generalizable results and stimulate overall progress in science for academics.

We provide clear theoretical arguments for the ability of GANs to generate artificial data that is equal to the data generating process in distribution. We test and provide evidence in favor of our methodology for three data sets in terms of the ability to generate high-quality artificial samples, predictive ability and the ability to derive meaningful insights from parameter estimates. Surprisingly, we show that the estimations on artificial data are able to outperform the estimations on real data in terms of predictive validity. Moreover, we compare different architectures of GANs from the AI literature to generate marketing data. Despite the identified difficulties with the convergence of GANs, we show how a deep convolutional generative adversarial network (DCGAN) and a generative adversarial network (GAN) are generally able to quickly generate high-quality samples of marketing data. We conclude with robustness checks to provide support for the decisions of the many hyperparameters in our architectural specifications.

7.2. Limitations

There are several limitations to this study. First of all, we are unable to account for the panel structure of the lemonade sales data. Therefore, we had to reduce the panel data to one supermarket chain and one lemonade brand (i.e., only 169 observations). This is considered to be an extremely limited amount in the application of GANs. One potential solution would be to treat the panel data set as we treat images in the AI literature. In the AI literature, we treat images as three-dimensional arrays. The first dimension refers to the rows, the second to the columns and the last dimension refers to the channels: red, green and blue pixels. To illustrate, we can transform the panel data set in a three-dimensional array where the first dimension denotes the period over which the data are recorded. The second dimension refers to the variables available in the data set and the last dimension refers to the individual dimensions. The individual dimension can be the supermarket chains or the lemonade brands. Unfortunately, the lemonade sales data set is unbalanced, therefore we are not able to treat the data in such a way. Another issue arises from the low number of observations in the lemonade sales data set. In case of image generation, the data usually consists of more than a million observations (Karras et al. 2017). Nevertheless, the results for the lemonade sales data set are still useful for marketing modeling. Therefore, we do show that the applications of GANs are not limited to data sets in the order of millions of observations.

Second, we obtain a correlation of zero when we compare the individual distributions of variables from p_G and p_{data} (see section 6.1). Therefore, we argue that the samples from p_G are not privacy sensitive. The literature has identified a large arrange of methods to evaluate whether artificial data are actually privacy-friendly. For example, Schneider et al. (2018) use the predictions of a multinomial logistic regression that the artificial data are unable to identify a unique store ID. The higher the probability that the artificial data are able to identify a unique store ID, the less privacy-friendly the data are. However, at the same time, we want our artificial data to be able to explain real-life events. Therefore, privacy protection and ability to derive meaningful insights is a trade-off that requires further investigation. Which brings us to future research in the next paragraph.

7.3. Future research

We identify several interesting avenues for future research. As we discussed, additional research is required to investigate the privacy - insights trade-off. One avenue to mitigate this trade-off would be to incorporate differential privacy into the training of a GAN (Abadi et al. 2016). Differential privacy has strong theoretical privacy guarantees (Dwork and Roth 2014). Differential privacy ensures that if all the components are differential private (e.g., each individual observation), then so is their composition. Therefore, it ensures a form of group privacy for the entire data set. Also, it is resilient to post-processing. For example, we would not be able to reverse engineer a GAN to output a sample from the p_{data} distribution. Less formal, differential privacy requires that no subject in the study has a significant influence on the information released by the algorithm. Therefore, we urge future research to incorporate differential privacy in a GAN.

Second, it is worthwhile to identify the effects of the generation of artificial data on the perceived privacy concerns among individuals. Martin et al. (2017) identify that privacy concerns (such as data breaches) among individuals reduce firm and industry performance. Transparency and trust positively mediate this effect. Therefore, firms would be able to use the proposed methodology to investigate the effects on privacy concerns. One could argue that our methodology creates transparency, because we are able to release the entire artificial data set. Therefore, why would an individual still have privacy concerns with regard to artificial data?

Third, the AI literature has a history of visualizing the learned representations present in neural networks. For example, we know that representations in image recognition networks resemble what our primary visual cortex registers when we look at images (Olshausen and Field 1996). Due to the composition of non-linear functions in a neural network, representations in earlier layers are easier to understand, but become increasingly abstract in later layers (Goodfellow et al. 2016). At the time of writing, we do not yet fully understand what representations are learned from the data in the context of marketing or more broadly economics and business. Radford et al. (2015) show that, due to learning representations, we are even able to manipulate the vector space to remove or add representations. Therefore, if the representations learned in a GAN resemble the underlying causes for the variation in the observed data from p_{data}, we might be able to manipulate the data to overcome econometric issues such as multicollinearity. Future research is required to identify what these representations consist of and how we are able to manipulate the vector space to overcome econometric or other data issues such as class imbalance.

Fourth, the field of *transfer learning* in the AI literature focuses on the ability to share a neural network from one prediction tasks to other prediction tasks assuming that the underlying causes for variation in the data are similar (Goodfellow et al. 2016). For example, the ability to predict churn and customer lifetime value (CLV). In this scenario, we are able to train a deep neural network to predict churn. The neural network learns relevant representations to explain variance in churn behavior. In the CLV prediction task, we train the neural network again on the second data set and use the trained neural network to predict CLV. As such, we are able to improve generalization to new data (Dauphin et al. 2012). Especially if the number of observations per class is low in the data for the second prediction task (Goodfellow et al. 2016). We are able to transfer neural networks to new prediction problems thanks to the learned representations that are able to explain variance in both prediction tasks. In some extreme cases, we do not even train the neural network again on the second data set to obtain a good prediction performance. This is also called zero-shot learning (Goodfellow et al. 2016).

After this little excursion into the AI literature, we return to the benefits of transfer learning to the marketing literature. We argue that marketing science could train one neural network and delete the data that it was trained on. Subsequently, instead of sharing data, we would be able to share the neural network with the learned representations to be trained on other data sets without violating the privacy of the customers in the data set. Especially, if we would train the neural network with differential privacy guarantees (Dwork and Roth 2014). The neural network would not only increase in predictive accuracy or generalize better over new observations, but we would also be able to preserve the privacy of the individual. We urge the literature to pursue such avenues to preserve the privacy of the individual while maintaining the ability to derive meaningful insights.

Finally, there are many more interesting potential applications of deep learning to marketing issues. For example, it would be interesting to investigate whether GANs are able to act as an imputation method to recover missing observations. Early evidence from Yoon et al. (2018) show that a GAN outperforms state-of-the-art imputation methods in terms of post-imputation predictions and the root mean squared error distance from the real observations. Another application would be to use the idea of adversarial examples from Goodfellow et al. (2014b) to reduce the generalization error in marketing prediction problems (see section 4.2). We hope that this study is the first to many more novel applications of deep learning in the field of marketing.

Acknowledgments

I would like to thank Prof. Jaap Wieringa gratefully for the thorough feedback, enthusiasm and patience during the process of writing this paper and support during the PhD application. I am looking forward to working together for the coming three years on the preservation of privacy in marketing during my PhD.

Furthermore, I would like to thank my parents, family and friends. Specifically some friends from the research master and future PhD colleagues who have helped me a great deal the last year: Steff Groefsema, Levi Kuiper, Hagen Kruse, Ryan Marapin and Jeroen van der Vaart.

References

Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L (2016) Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16* URL http://dx.doi.org/10.1145/2976749.2978318.

Acquisti A, Brandimarte L, Loewenstein G (2015) Privacy and human behavior in the age of information. *Science* 347(6221):509–514, ISSN 0036-8075, URL http://dx.doi.org/10.1126/science.aaa1465.

Acquisti A, Taylor C, Wagman L (2016) The Economics of Privacy. *Journal of Economic Literature* 54(2):442–492, URL https://ideas.repec.org/a/aea/jeclit/v54y2016i2p442-92.html.

Arjovsky M, Chintala S, Bottou L (2017) Wasserstein gan.

Badia AP, Piot B, Kapturowski S, Sprechmann P, Vitvitskyi A, Guo D, Blundell C (2020) Agent57: Outperforming the atari human benchmark.

Beaufays F (2015) The neural networks behind google voice transcription. URL https://ai.googleblog.com/2015/08/the-neural-networks-behind-google-voice.html.

Beaulieu-Jones BK, Wu ZS, Williams C, Lee R, Bhavnani SP, Byrd JB, Greene CS (2019) Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes* 12(7):e005122, URL http://dx.doi.org/10.1161/CIRCOUTCOMES.118.005122.

Bengio Y (2009) Learning deep architectures for ai. *Found. Trends Mach. Learn.* 2(1):1–127, ISSN 1935-8237, URL http://dx.doi.org/10.1561/2200000006.

Bengio Y, Frasconi P, Simard P (1993) The problem of learning long-term dependencies in recurrent networks. *IEEE International Conference on Neural Networks*, 1183–1188 vol.3.

Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5(2):157–166.

Biau G, Cadre B, Sangnier M, Tanielian U (2018) Some theoretical properties of gans.

Billingsley P (1986) *Probability and Measure* (John Wiley and Sons), second edition.

Bishop CM (2006) *Pattern Recognition and Machine Learning (Information Science and Statistics)* (Berlin, Heidelberg: Springer-Verlag), ISBN 0387310738.

Breiman L (2001) Random forests. *Mach. Learn.* 45(1):5–32, ISSN 0885-6125, URL http://dx.doi.org/10.1023/A:1010933404324.

Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler DM, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D (2020) Language models are few-shot learners.

Bucklin RE, Sismeiro C (2009) Click here for internet insight: Advances in clickstream data analysis in marketing. *Journal of Interactive Marketing* 23(1):35 – 48, ISSN 1094-9968, URL `http://dx.doi.org/https://doi.org/10.1016/j.intmar.2008.10.004`, anniversary Issue.

Choromanska A, Henaff M, Mathieu M, Arous GB, LeCun Y (2014) The loss surface of multilayer networks. *CoRR* abs/1412.0233, URL `http://arxiv.org/abs/1412.0233`.

Columbus L (2014) 2014: The year big data adoption goes mainstream in the enterprise. URL `https://www.forbes.com/sites/louiscolumbus/2014/01/12/2014-the-year-big-data-adoption-goes-mainstream-in-the-enterprise/`.

Danaher PJ, Smith MS (2011) Modeling multivariate distributions using copulas: Applications in marketing. *Marketing Science* 30(1):4–21, URL `http://dx.doi.org/10.1287/mksc.1090.0491`.

Dauphin GMY, Glorot X, Rifai S, Bengio Y, Goodfellow I, Lavoie E, Muller X, Desjardins G, Warde-Farley D, Vincent P, Courville A, Bergstra J (2012) Unsupervised and transfer learning challenge: a deep learning approach. Guyon I, Dror G, Lemaire V, Taylor G, Silver D, eds., *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, 97–110 (Bellevue, Washington, USA: PMLR), URL `http://proceedings.mlr.press/v27/mesnil12a.html`.

Dauphin YN, Pascanu R, Gulcehre C, Cho K, Ganguli S, Bengio Y (2014) Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ, eds., *Advances in Neural Information Processing Systems 27*, 2933–2941 (Curran Associates, Inc.), URL `http://papers.nips.cc/paper/5486-identifying-and-attacking-the-saddle-point-problem-in-high-dimensional-non-convex-optimizatic pdf`.

Dwork C, Roth A (2014) The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9(3–4):211–407, ISSN 1551-305X, URL `http://dx.doi.org/10.1561/0400000042`.

European Commission (2012) Regulation of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (general data protection regulation).
`https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52012PC0011`.

Fawzi A, Moosavi-Dezfooli S, Frossard P, Soatto S (2017) Classification regions of deep neural networks. *CoRR* abs/1705.09552, URL `http://arxiv.org/abs/1705.09552`.

Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. Teh YW, Titterington M, eds., *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, 249–256 (Chia Laguna Resort, Sardinia, Italy: PMLR), URL `http://proceedings.mlr.press/v9/glorot10a.html`.

Goldfarb A, Tucker CE (2011) Privacy regulation and online advertising. *Management Science* 57(1):57–71, URL `http://dx.doi.org/10.1287/mnsc.1100.1246`.

Goodfellow I (2016) Nips 2016 tutorial: Generative adversarial networks.

Goodfellow I, Bengio Y, Courville A (2016) *Deep Learning* (MIT Press), `http://www.deeplearningbook.org`.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014a) Generative adversarial nets. Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ, eds., *Advances in Neural Information Processing Systems 27*, 2672–2680 (Curran Associates, Inc.), URL `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

Goodfellow IJ, Bulatov Y, Ibarz J, Arnoud S, Shet V (2013) Multi-digit number recognition from street view imagery using deep convolutional neural networks.

Goodfellow IJ, Shlens J, Szegedy C (2014b) Explaining and harnessing adversarial examples.

Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J (2009) A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(5):855–868, URL `http://dx.doi.org/10.1109/TPAMI.2008.137`.

Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A (2017) Improved training of wasserstein gans.

Hauser M, Ray A (2017) Principles of riemannian geometry in neural networks. Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, eds., *Advances in Neural Information Processing Systems 30*, 2807–2816 (Curran Associates, Inc.), URL `http://papers.nips.cc/paper/6873-principles-of-riemannian-geometry-in-neural-networks.pdf`.

Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Computation* 9(8):1735–1780.

Holtrop N, Wieringa JE, Gijsenberg MJ, Verhoef PC (2017) No future without the past? predicting churn in the face of customer privacy. *International Journal of Research in Marketing* 34(1):154 – 172, ISSN 0167-8116, URL `http://dx.doi.org/https://doi.org/10.1016/j.ijresmar.2016.06.001`.

Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift.

Karras T, Aila T, Laine S, Lehtinen J (2017) Progressive growing of gans for improved quality, stability, and variation.

Khaitan P (2016) Chat smarter with allo. URL `https://ai.googleblog.com/2016/05/chat-smarter-with-allo.html`.

Kingma DP, Ba J (2014) Adam: A method for stochastic optimization.

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Pereira F, Burges CJC, Bottou L, Weinberger KQ, eds., *Advances in Neural Information Processing Systems 25*, 1097–1105 (Curran Associates, Inc.), URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

Kumar A, Biswas A, Sanyal S (2018) ecommercegan : A generative adversarial network for e-commerce.

LeCun Y, Bottou L, Orr GB, Müller KR (1998) Efficient backprop. *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, 9–50 (Berlin, Heidelberg: Springer-Verlag), ISBN 3540653112.

Leeflang P, Bijmolt T, Pauwels K, Wieringa J (2015) *Modeling Markets: Analyzing Marketing Phenomena and Improving Marketing Decision Making*. International Series in Quantitative Marketing (Springer), ISBN 978-1-4939-2086-0, URL `http://dx.doi.org/10.1007/978-1-4939-2086-0`.

Leeflang P, Wieringa J, Bijmolt T, Pauwels K, eds. (2017) *Advanced Methods for Modeling Markets*. International Series in Quantitative Marketing (Springer), ISBN 978-3-319-53467-1.

Lemmens A, Croux C (2006) Bagging and boosting classification trees to predict churn. *Journal of Marketing Research* 43(2):276–286, URL `http://dx.doi.org/10.1509/jmkr.43.2.276`.

Leshno M, Lin VY, Pinkus A, Schocken S (1993) Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks* 6(6):861 – 867, ISSN 0893-6080, URL `http://dx.doi.org/https://doi.org/10.1016/S0893-6080(05)80131-5`.

Lin J (1991) Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory* 37(1):145–151, ISSN 0018-9448, URL `http://dx.doi.org/10.1109/18.61115`.

Marketing Science Institute (2020) *RESEARCH PRIORITIES 2020-2022*.

Martin KD, Borah A, Palmatier RW (2017) Data privacy: Effects on customer and firm performance. *Journal of Marketing* 81(1):36–58, URL `http://dx.doi.org/10.1509/jm.15.0497`.

Mathews L (2017) Equifax data breach impacts 143 million americans. URL `https://www.forbes.com/sites/leemathews/2017/09/07/equifax-data-breach-impacts-143-million-americans/#6f6ed8d3356f`.

McLean R (2018) Quora says 100 million users hit by 'malicious' data breach. URL `https://edition.cnn.com/2018/12/03/tech/quora-hack-data-breach/index.html`.

Miller AR, Tucker CE (2011) Encryption and the loss of patient data. *Journal of Policy Analysis and Management* 30(3):534–556, URL `http://dx.doi.org/10.1002/pam.20590`.

Neate R (2018) Over $119bn wiped off facebook's market cap after growth shock. URL `https://www.theguardian.com/technology/2018/jul/26/facebook-market-cap-falls-109bn-dollars-after-growth-shock`.

Olshausen B, Field D (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381:607–9, URL `http://dx.doi.org/10.1038/381607a0`.

Peltier JW, Zahay D, Lehmann DR (2013) Organizational learning and crm success: A model for linking organizational practices, customer data quality, and performance. *Journal of Interactive Marketing* 27(1):1 – 13, ISSN 1094-9968, URL `http://dx.doi.org/https://doi.org/10.1016/j.intmar.2012.05.001`.

Radford A, Metz L, Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks.

Raeesy Z, Gillespie K, Ma C, Drugman T, Gu J, Maas R, Rastrow A, Hoffmeister B (2018) Lstm-based whisper detection. *CoRR* abs/1809.07832, URL `http://arxiv.org/abs/1809.07832`.

Reiter J (2010) Multiple imputation for disclosure limitation: Future research challenges. *Journal of Privacy and Confidentiality* 1(2), URL `http://dx.doi.org/10.29012/jpc.v1i2.575`.

Reuters (2019) Facebook may face multibillion-dollar us fine over privacy lapses – report. URL `https://www.theguardian.com/technology/2019/feb/14/facebook-ftc-privacy-cambridge-analytica-fine`.

Risselada H, Verhoef P, Bijmolt T (2010) Staying power of churn prediction models. *Journal of Interactive Marketing - J INTERACT MARK* 24:198–208, URL `http://dx.doi.org/10.1016/j.intmar.2010.04.002`.

Rome S (2017) An annotated proof of generative adversarial networks with implementation notes. URL `https://srome.github.io/An-Annotated-Proof-of-Generative-Adversarial-Networks-with-Implementation-Notes/`.

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. *Nature* 323:533–536.

Rust RT (2019) The future of marketing. *International Journal of Research in Marketing* ISSN 0167-8116, URL `http://dx.doi.org/https://doi.org/10.1016/j.ijresmar.2019.08.002`.

Salakhutdinov R, Hinton G (2007) Learning a nonlinear embedding by preserving class neighbourhood structure. Meila M, Shen X, eds., *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, 412–419 (San Juan, Puerto Rico: PMLR), URL `http://proceedings.mlr.press/v2/salakhutdinov07a.html`.

Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X (2016) Improved techniques for training gans.

Schneider MJ, Jagpal S, Gupta S, Li S, Yu Y (2017) Protecting customer privacy when marketing with second-party data. *International Journal of Research in Marketing* 34(3):593–603, URL `http://dx.doi.org/10.1016/j.ijresmar.2017.0`.

Schneider MJ, Jagpal S, Gupta S, Li S, Yu Y (2018) A flexible method for protecting marketing data: An application to point-of-sale data. *Marketing Science* 37(1):153–171, URL `http://dx.doi.org/10.1287/mksc.2017.1064`.

Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap T, Leach M, Kavukcuoglu K, Graepel T, Hassabis D (2016) Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489, ISSN 1476-4687, URL `http://dx.doi.org/10.1038/nature16961`.

Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, Chen Y, Lillicrap T, Hui F, Sifre L, van den Driessche G, Graepel T, Hassabis D (2017) Mastering the game of go without human knowledge. *Nature* 550(7676):354–359, ISSN 1476-4687, URL `http://dx.doi.org/10.1038/nature24270`.

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958, URL `http://jmlr.org/papers/v15/srivastava14a.html`.

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* URL `http://dx.doi.org/10.1109/cvpr.2016.308`.

Theis L, van den Oord A, Bethge M (2015) A note on the evaluation of generative models.

Tomasev N, Glorot X, Rae JW, Zielinski M, Askham H, Saraiva A, Mottram A, Meyer C, Ravuri S, Protsyuk I, Connell A, Hughes CO, Karthikesalingam A, Cornebise J, Montgomery H, Rees G, Laing C, Baker CR, Peterson K, Reeves R, Hassabis D, King D, Suleyman M, Back T, Nielson C, Ledsam JR, Mohamed S (2019) A clinically applicable approach to continuous prediction of future acute kidney injury. *Nature* 572(7767):116–119, ISSN 1476-4687, URL `http://dx.doi.org/10.1038/s41586-019-1390-1`.

van den Oord A, Dambre J (2015) Locally-connected transformations for deep gmms. *International Conference on Machine Learning (ICML) : Deep learning Workshop, Abstracts*, 1–8, URL `https://sites.google.com/site/deeplearning2015/20.pdf?attredirects=0`.

van Doorn J, Hoekstra JC (2013) Customization of online advertising: The role of intrusiveness. *Marketing Letters* 24(4):339–351, ISSN 1573-059X, URL `http://dx.doi.org/10.1007/s11002-012-9222-1`.

Wedel M, Kannan P (2016) Marketing analytics for data-rich environments. *Journal of Marketing* 80(6):97–121, URL `http://dx.doi.org/10.1509/jm.15.0413`.

Wieringa J, Kannan P, Ma X, Reutterer T, Risselada H, Skiera B (2019) Data analytics in a privacy-concerned world. *Journal of Business Research* ISSN 0148-2963, URL `http://dx.doi.org/https://doi.org/10.1016/j.jbusres.2019.05.005`.

Williams CKI (1997) Computing with infinite networks. Mozer MC, Jordan MI, Petsche T, eds., *Advances in Neural Information Processing Systems 9*, 295–301 (MIT Press), URL `http://papers.nips.cc/paper/1197-computing-with-infinite-networks.pdf`.

Wilson DR, Martinez TR (2003) The general inefficiency of batch training for gradient descent learning. *Neural Netw.* 16(10):1429–1451, ISSN 0893-6080, URL `http://dx.doi.org/10.1016/S0893-6080(03)00138-2`.

Yoon J, Jordon J, van der Schaar M (2018) Gain: Missing data imputation using generative adversarial nets.

Zynga (2019) Player security announcement. URL `https://investor.zynga.com/news-releases/news-release-details/player-security-announcement`.