Online Appendix: Using Deep Learning to Overcome Privacy and Scalability Issues in Customer Data Transfer*

Piyush Anand

Clarence Lee

March 24, 2022

*Piyush Anand is Assistant Professor of Marketing, Jones Graduate School of Business, Rice University (piyush.anand@rice.edu), Clarence Lee is Co-Founder at Eisengard AI (clarence@eisengard.ai)

Appendix A Online Appendix

A.1 Inference Model and Information Loss

We now describe information loss and the inference model. As an example of a multiple regression framework with continuous independent and dependent variables, consider the following log-log regression in a standard panel data setting with entity i, brand j, and time period t

$$lnY_{ijt} = \delta_j + \beta_j lnX_{ijt} + \sum_{k \neq j} \beta_k lnX_{ikt} + \epsilon_{ijt}, \tag{A.1}$$

for k number of covariates of interest, where δ_j s the brand-level intercept. This log-log regression framework has been used widely in marketing and economics, modeling continuous dependent variables such as store sales, worker wages, and customer demand

One context that falls under this framework is the market response model proposed in SCAN*PRO (Leeflang and Wittink, 2000), which has been used extensively by AC Nielsen and consumer goods manufacturers (Schneider et al., 2018). The SCAN*PRO model uses a log-log framework with own and competitor pricing as covariates and sales as the dependent variable.[1] This corresponds to a data generating process similar to a nonlinear customer response function, such as the multiplicative specification proposed by (Wittink et al., 1988) for store-level data.

Therefore, we formally capture weekly customer sales for multiple brands:

$$S_{ijt} = \alpha_{ij} P_{ijt}^{\beta_j} \prod_{k \neq j} P_{ikt}^{\beta_k} e^{\epsilon_{ijt}} \ , i = 1, ..., n; t = 1, ..., T, \tag{A.2}$$

which translates to a log-log regression model:

$$lnS_{ijt} = \mu_j + \mu_{ij} + \beta_j lnP_{ijt} + \sum_{k \neq j} \beta_k lnP_{ikt} + \epsilon_{ijt}, \tag{A.3}$$

where S_{ijt} denotes household weekly purchase in dollar amount and P_{ijt} price in dollars for customer i, product brand j, and week t. P_{ikt} are all prices observed by the customer for competitor brands; μ_j is the brand specific intercept term and μ_{ij} the household-level random effects term drawn from a normal distribution $N(0, \sigma_\mu)$. For log sales models in which the independent variables are log-prices, β_j, the price coefficient, is also the price elasticity; ϵ is the unobserved, independent error term.

Mean absolute percentage difference (MAPD), proposed by Christen et al. (1997) as a measure of the difference between the regression estimate of a market-level aggregation method and true data, allows for assessing the relative difference between a protection method and true data. Essentially, MAPD measures the average absolute difference of coefficient estimates across J number of coefficients of interest.

$$MAPD = \frac{1}{J} \sum_{j=1}^{J} \left| \frac{\hat{\beta}_j - \beta_j}{\beta_j} \right| \ \times \ 100\%, \tag{A.4}$$

where $\hat{\beta}_j$ is the estimated coefficient of interest on protected, and β_j the estimated coefficient on real data and J refers to the number of relevant coefficients to be analyzed using a statistical

[1] We exclude promotion indicator variables in this proof of concept in order to focus on the method's power to protect continuous independent and dependent variables. We will explore how to protect indicator variables in future research.

modeling technique (e.g., regression). We use the brands' own price elasticities as the coefficients of interest in the subsequent sections when MAPD is reported.

A.2 Monte Carlo Experiment 1 Data

In this section, we describe the Monte Carlo data generation process for experiment 1. We use this data to benchmark GANs and other data protection models. We use the following equation (from section A.1) to generate the sales data S for 5 brands (j), 200 customers(i), and 52 weeks (t):

$$lnS_{ijt} = \mu_j + \delta_{ij} + \beta_j lnP_{ijt} + \epsilon_{ijt}$$

We chose the values for the parameters: $\mu = (-0.1, -0.05, -0.09, -0.3, -0.1)$ is inherent preference for a brand j, $\delta = N(0, 2)$ is the customer specific random effect i for a brand j. For the price elasticities, we borrow from Christen et al. (1997) the price elastitices: $\beta = (-1.5, -1.7, -2.01, -1.98, -1.9)$. We draw prices for a brand j as random draws from a normal distribution as follows: $P_j = N(\mu p_j, \sigma p_j)$, where $\mu p = (8.68, 5.09, 5.48, 5.94, 4.15)$ and $\sigma p = (0.93, 0.5, 0.5, 0.39, 0.77)$.

With the above parameter settings, we generate the customer sales data that is subsequently used as the "true" data for the Monte Carlo data.

We also examine the proposed GAN's generated synthetic data distributional accuracy relative to that of the true data. Figure A.1 compares data densities for log-sales of all brands protected by various methods. For purposes of this analysis, the 20% and 50% swap methods will not provide meaningful comparisons, as the distribution of the variable of interest (sales), by construction, does not change after merely swapping sales in the data.[2]

Across the five brands, we find that GANs fit the true data the most closely, and that the fit for GAN (Het.) is higher than for GAN (No Het.). The next best fit is from top-coding, which by construction, differs from true data post the truncation point of 95th percentile. We also find that random noise and rounding fluctuate around the true data, and have a poorer fit.

[2]Nor would market level data be considered, as this analysis aggregates data across customers at the weekly level and the distribution range would not be amenable to comparison using methods that track individual customer level sales.

Figure A.1: Distributional Accuracy for Synthetic Data

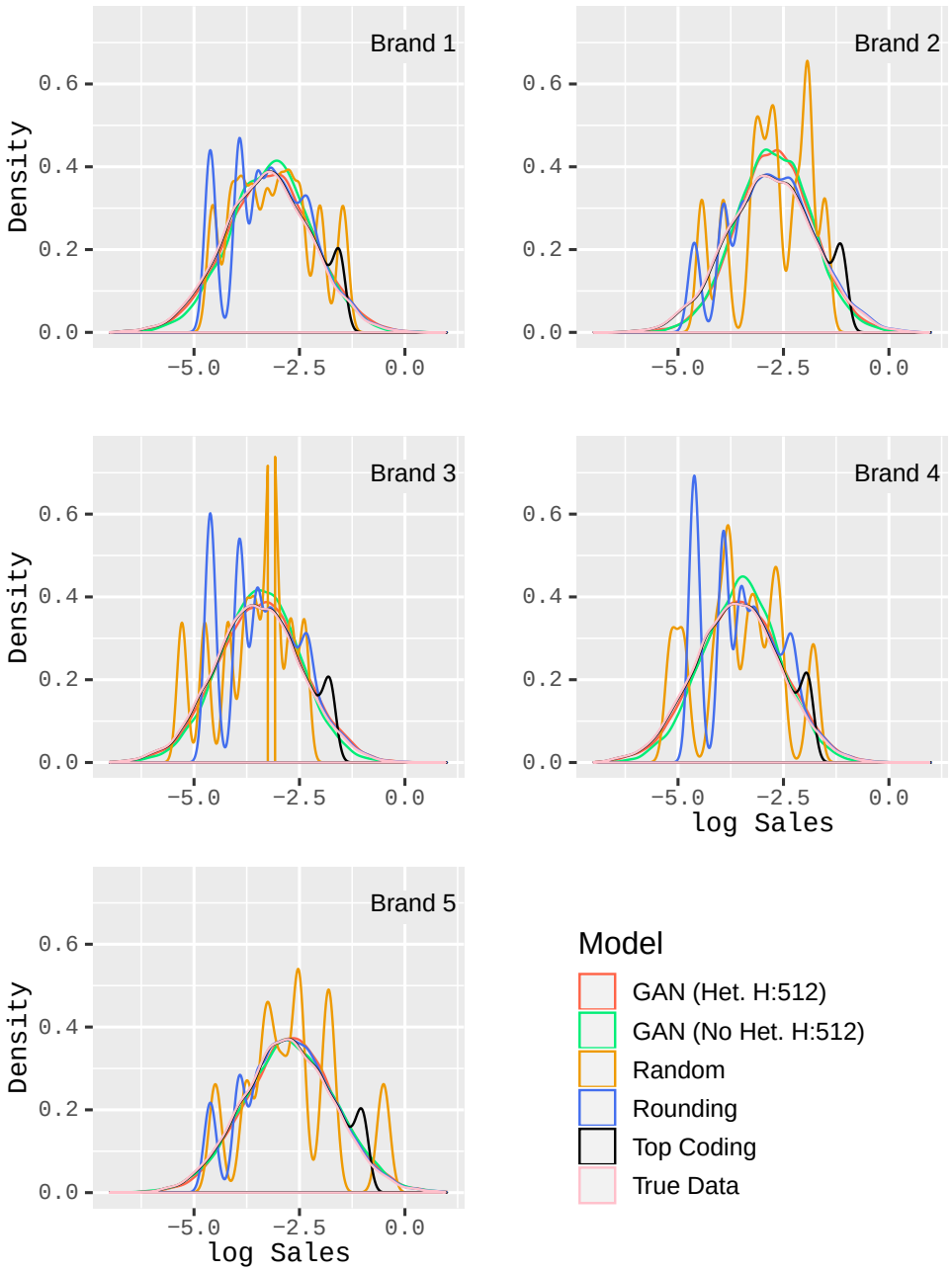


Table A.1: Summary Statistics for Monte Carlo Data (N=10,400)

This table shows the summary statistics for the Monte Carlo data generated using the process described in the Online Appendix A.2. Variable Visibility column lists whether a variable is public data or private data. The prices are public data, i.e. observed by both researcher, and the sales are private data, i.e. visible only to the firm. Note that the generator in the GAN model has access to only the public data (prices), and the private data (sales) never leaves the walls of the firm.

Variable	Mean	Std. Dev.	Min	Max	Variable Visibility
Price (Brand 1)	8.69	0.93	5.06	12.24	Public
Price (Brand 2)	5.08	0.50	2.92	7.11	Public
Price (Brand 3)	5.48	0.50	3.51	7.34	Public
Price (Brand 4)	5.94	0.39	4.37	7.45	Public
Price (Brand 5)	4.15	0.77	0.90	7.40	Public
Sales (Brand 1)	0.07	0.09	0	1.40	Private
Sales (Brand 2)	0.11	0.15	0	3.75	Private
Sales (Brand 3)	0.05	0.09	0	4.38	Private
Sales (Brand 4)	0.05	0.06	0	1.21	Private
Sales (Brand 5)	0.12	0.19	0	4.87	Private

A.3 Monte Carlo Experiment 2 - Customer Targeting

We construct purchase behavior for 30 customers and 365 days building on the purchase probability model from Park and Park (2016). More formally:

$$purchase^*_{i,t} = \alpha + \gamma Minority_i + \beta_v Visits_{i,t} + \beta_m Marketing_{i,t} + \beta_a Age_i + \beta_i Income_i + \beta_w Weight_i + \delta PreviousPurchase_{i,t}$$

$$purchase_{i,t} = 1 \;\; if \;\; purchase^*_{i,t} > 0 \tag{A.5}$$

where $purchase_{i,t}$ is whether a customer i makes a purchase on a day t. The variables that we include directly from Park and Park (2016) in the customer's utility function are: $Visits_{i,t}$ is the log of the visits made by the customers to store so far, $Marketing_{i,j}$ is a dummy for whether a customer was marketed on the day t or not, and $PreviousPurchase, i, t$ is the dummy for whether the customer made a purchase on the preceding day t. We borrow the coefficients from Park and Park (2016). In addition to these variables, we add customer demographic variables: Age_i is the log of the age of the customer in number of years, $Income_i$ is the log of the income (in thousands) of the customer, $Weight_i$ is the log of the weight (in pounds) of the customer, and $Minority_i$ is a dummy for whether the customer belongs to a racial minority or not. ϵ_{ij} is the random error term drawn from type I extreme value distribution. The coefficients for the data are: $\alpha = -2.547, \gamma = -0.273, \beta_v = 0.269, \beta_m = 1.210, \beta_a = 0.51, \beta_i = 0.5, \beta_w = -0.2, \delta = -0.678$.We borrow the coefficients for previous purchase, marketing, and visits so far from Park and Park (2016). We consider the data constructed in this manner as the true data.

Table A.2 shows the summary statistics for the data. Figure 5(a) in the main paper shows the results for GANs (and benchmarks) for customer targeting using the data generated in this manner.

Table A.2: Summary Statistics for Monte Carlo Data (N=10,950)

This table shows the summary statistics for the Monte Carlo data generated using the process described in Section 3.3. Variable Visibility column lists whether a variable is public data or private data. The following variables are public: Marketing, Previous Purchase, and log(Visits So Far), i.e. observed by both researcher, and the Purchase variable is private data, i.e. visible only to the firm. Note that the generator in the GAN model has access to only the public data, and the private data never leaves the walls of the firm.

Variable	Mean	Std. Dev.	Min	Max	Variable Visibility
log(Age)	4.00	0.12	3.74	4.17	Public
log(Income)	3.73	0.16	3.53	3.99	Public
Marketing	0.10	0.29	0	1.00	Public
Previous Purchase	0.13	0.34	0	1.00	Public
log(Visits so Far)	2.21	0.81	0	3.58	Public
log(Weight)	5.46	0.14	5.22	5.68	Public
Minority	0.27	0.44	0	1.00	Private
Purchase	0.13	0.34	0	1.00	Private

We first report the distributional summary statistics for the protected variables of purchase and minority obtained from GANs, and other benchmarks, as compared to the real data.[3]

[3]The variables of purchase and minority are binary variables and are obtained by transforming the GAN output to "discrete" output, which is different from the "continuous" data contexts in the other monte carlo datasets. We use K-means clustering to transform the output of GANs to discrete, 0 and 1 outputs for purchase and minority variables. K-means clustering and softmax classification are related approaches (Hess et al., 2020). Other activation

We report the mean and standard deviation for the variables of purchase and minority in Table A.3. Note that Swap 20 and Swap 50 benchmarks will not change the distributional statistics for these variables by definition, since these are swapping random subsets (20% and 50% respectively) within the data. Additionally, we also report the conditional statistics for minority (conditional on purchase = 0 and on purchase =1) in Table A.4, and conditional statistics for purchase (conditional on minority = 0 and on minority = 1) in Table A.5. We find that the protected variables of purchase and minority generated by GANs are close to the real data in both the summary statistics and conditional summary statistics.

Table A.3: Summary Statistics for Protected Variables: Purchase and Minority

	Minority		Purchase	
Model	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**
Real Data	0.27	0.44	0.13	0.34
GAN	0.28	0.45	0.16	0.37
Swap 20	0.27	0.44	0.13	0.34
Swap 50	0.27	0.44	0.13	0.34

Table A.4: Summary Statistics for Protected Variable: Minority Conditional on Purchase

	Minority | Purchase = 0		Minority | Purchase = 1	
Model	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**
Real Data	0.25	0.43	0.41	0.49
GAN	0.26	0.44	0.38	0.48
Swap 20	0.25	0.43	0.38	0.49
Swap 50	0.26	0.44	0.33	0.47

Table A.5: Summary Statistics for Protected Variable: Purchase Conditional on Minority

	Purchase | Minority = 0		Purchase | Minority = 1	
Model	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**
Real Data	0.11	0.31	0.2	0.4
GAN	0.14	0.35	0.22	0.42
Swap 20	0.11	0.31	0.19	0.39
Swap 50	0.12	0.33	0.17	0.37

functions and approaches can also be used to obtain discrete outputs from GANs and this along with other GAN architectures can be explored in future research.

Additionally, here we explore three key differences between the firms' inference model and true data-generating process to test for robustness of GANs:

1. **Difference in missing variables:** what happens if the firms' models has variables that are missing from the true DGP, especially if the X variables are correlated?

2. **Difference in error and heterogeneity:** what happens to the firms' models if the error in the DGP is large and/or if there are existing heterogeneity in the data?

3. **Difference in functional form:** what happens if the functional form of the inference model is different from true DGP?

First, we explore what happens if there are missing data in the firms' inference models. We exclude a variable from being shared with the researcher, even when it is a part of the data generating process. When evaluating GANs and other benchmarks, the variable "weight" is not shared, i.e. GANs training data does not include the variable weight, the benchmarks are also not provided with this variable. This corresponds to the "+ Missing Data" set of results in Figure A.2. Note that the baseline specification (same as Figure 5b) is the "*" Standard Specification in Figure A.2.

Next, we estimate with correlated explanatory variables. Note that the baseline condition has independent explanatory (X) variables, we next test for robustness with correlated explanatory variables. We set the covariance matrix for the variables of age, income, and weight as:

3.21	-2.54	4.95
-2.54	4.95	1.47
4.95	1.47	1.62

The "+ Correlated X" set in Figure A.2 shows results for correlated data. We estimated different levels of co-variances, and found that GANs consistently outperform on the accuracy-privacy trade-off. For brevity, we do not include those results here.

Next, we explore heterogeneity in the effects. For this - we explore two levels of heterogeneity based on random effects coefficients for each customer. For the "Heterogeneity Low", i.e. low heterogeneity case, the coefficients for each customer are drawn from normal distributions with the following mean and variance: $\alpha = N(-2.547, 0.641), \gamma = N(-0.273, 0.321), \beta_v = N(0.269, 0.165), \beta_m = N(1.210, 0.273), \beta_a = N(0.51, 0.005), \beta_i = N(0.5, 0.002), \beta_w = N(-0.2, 0.04)$. For the "Heterogeneity High", i.e. high heterogeneity case, we use variances that are four times the variances in the "Low" heterogeneity case.

We then explore the effect of non-linear terms in the data generating process. For this, we add to the data generating process a squared term for customer visits so far, with the coefficient $\beta_{v2} = -0.2$. This allows for non-linear association between customer purchase and customer visits. These correspond to the "+ Non Linear DGP" set of results.

In addition to differences in the variable space, we also explore difference in error term in the data generating process. Note that in the standard specification, the error is drawn from a Gumbel with size = 0 and scale = 1. Now, we construct synthetic data with the error term drawn from a Gumbel type 1 EV distribution with scale of 1.05 and 0.95. Note that changing the scale parameter of the Gumbel distribution shifts both the mean and variance of the distribution. As a result, the percentage of customers-days where we observe a purchase changes from 13% in the baseline condition to 11% in the low scale condition and 15% in the high scale condition. This is because both the error mean and variance change due to change in the scale of the Gumbel distribution - see

Table A.6, and these push the customers at the margin to make the purchase (or not) depending on the error term. We find that in both these conditions, the GAN's loss in accuracy is substantially similar - from the baseline 0.13% loss in information to 0.17% in the low error scale condition and 0.10% in the high error scale condition. The findings for accuracy-privacy implications for GANs are similar across the three error scale conditions. These results are reassuring that GANs provide robustness to different error sizes. We do not report these in the figure here for brevity and clarity.

Table A.6: DGP: With Different Error Scale

size	scale	mean	variance	purchase	Loss in Information: GAN (1 - F1)
0	0.95	0.55	1.48	0.11	0.17
0	1	0.58	1.64	0.13	0.13
0	1.05	0.61	1.81	0.15	0.10

We next examine GANs effectiveness with differences in the researcher's inference model and that of the true data generating process. We do this in two steps. In the first step, we apply the data protection methods to generate data using GANs and benchmarks. In the second step, we split our data on independent variables into a training set (95% random subset) and a test set (5% random subset) of the data. The outcome data (purchase) is also split in the same process, but we are only interested in comparing predictions with the true outcome, i.e. purchase in the real data to compute loss in information. For consistency, we keep the same observations (X values) in the test set when comparing different data protection methods. We then train Random Forests on the training set of these independent and outcome variable: Purchase (Real) for Real Data, Purchase (GANs) for GANs, Purchase (Swap 20) for Swap 20, and Purchase (Swap 50) for Swap 50. Therefore, the Random Forest trained on the real data observes training set of independent variables and purchase data from the real data, whereas the Random Forest with GANs observes training set of independent variables and purchase data generated using GANs, and similarly for Swap 20 and Swap 50. We finally obtain the predictions for the test set of data from the different Random Forests and compare them to the test outcome - purchase data in a relative manner. That is, we compute information loss (1-F1) for each of the Random Forests, and report the relative information Loss. Information Loss (Relative) is defined as the additional loss in information from using GANs and benchmarks, as opposed to using real data.

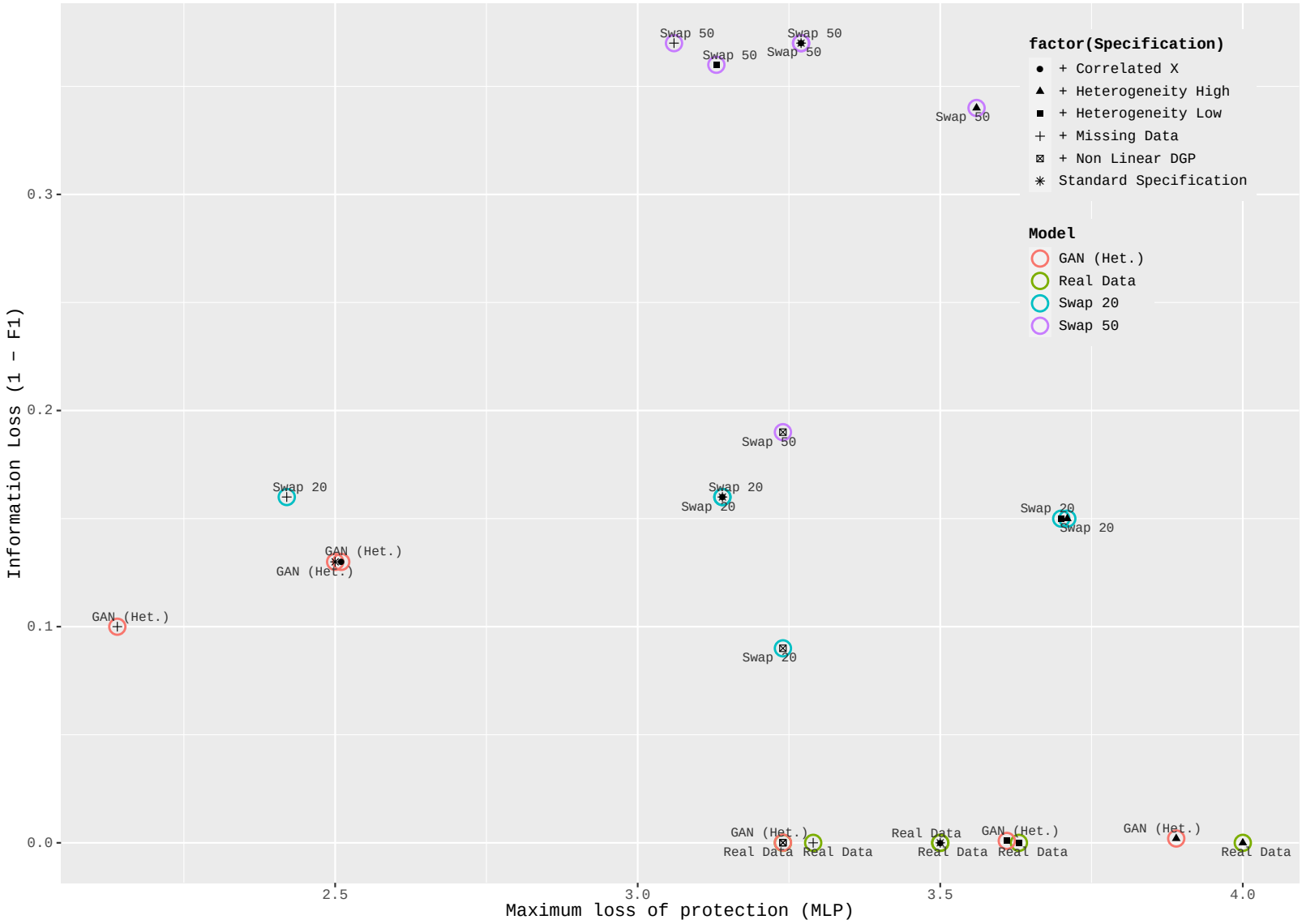
To operationalize this, we train a standard Random Forest based classifier with 100 decision trees,[4] on 95% of data, i.e. with real data, benchmarks, and GANs. We then predict with these random forests on 5% test data and report the additional loss of of information (1-F1) for Random Forests for GANs and benchmarks, as compared to the loss of information (1-F1) for Random Forest trained on the real data. Information Loss (Relative) is thus the incremental loss of predictive power in Random Forests trained on GANs and benchmarks (as compared to Random Forest trained on the real data). Table A.7 shows the results. These results are reassuring that GANs provide robustness to different inference models.

Table A.7: Robustness to Researcher's Model: Random Forests for Inference

Model	Loss of Privacy (MLP)	Information Loss (Relative)
GAN	2.5	0.01
Swap 20	3.14	0.09
Swap 50	3.27	0.16

[4]We use sklearn RandomForestClassifier package in python to estimate Random Forests

Figure A.2: Robustness Results with Different Data Specifications



Finally, we test GANs (and Benchmarks) on larger data in Section 5.3.2. We estimate the data for 3,000 customers for 365 days, for a total of 1.095 Million observations. Table A.8 shows the results. We find that GANs outperform even with larger data volumes.

Table A.8: Robustness to Large Data Volume

Model	**Loss of Privacy (MLP)**	**Information Loss (1 - F1)**
Real Data	3.6	0
Swap 20	3.6	0.16
Swap 50	3.49	0.36
GAN	3.35	0.02

In summary, through the above variations in the data and inference models, we find that GANs are a suitable substitute for traditional data obfuscation benchmarks. Not surprisingly, we find that GANs have a lower loss in information when there is heterogeneity and non-linearity in the data. This is because as GANs, based on neural networks, can exploit such data better. Also unsurprisingly, we find that the loss in privacy is higher in these contexts, as customer heterogeneity and non-linearities in the data make customer identity easier to identify.

A.4 Monte Carlo Experiment 3 Data - Tackling Multiple Marketing Problems

The data generating process specification is along the lines of Schneider et al. (2018). More formally:

$$lnS_{ijt} = \mu_j + \mu_{ij} + \beta_j lnP_{ijt} + ln(\delta_{fj})F_{ijt} + ln(\delta_{dj})D_{ijt} + ln(\delta_{fdj})FD_{ijt} + \epsilon_{ijt}, \tag{A.6}$$

The price distribution and coefficients are the same as those described in the Online Appendix A.2. The feature, display, and feature and display coefficients are from Schneider et al. (2018). In order to simulate whether a brand was featured, displayed, or both for a customer-brand-week, we draw randomly from a uniform distribution, with the following thresholds for feature: $(0.3, 0.2, 0.15, 0.25, 0.1)$, and the following threshold for display: $(0.25, 0.15, 0.05, 0.15, 0.1)$.[5] Thus, with this data generating process, we construct a dataset of 200 customers for 52 weeks and 5 brands, and compare the MAPD GANs, as compared to benchmarks, for price coefficients, and marketing variables coefficients for feature, display, and both.

[5]We note that the choice of these thresholds is to illustrate as a proof of concept, and that these thresholds should not influence the subsequent results.

Table A.9: Summary Statistics for Monte Carlo Data (N=10,400)

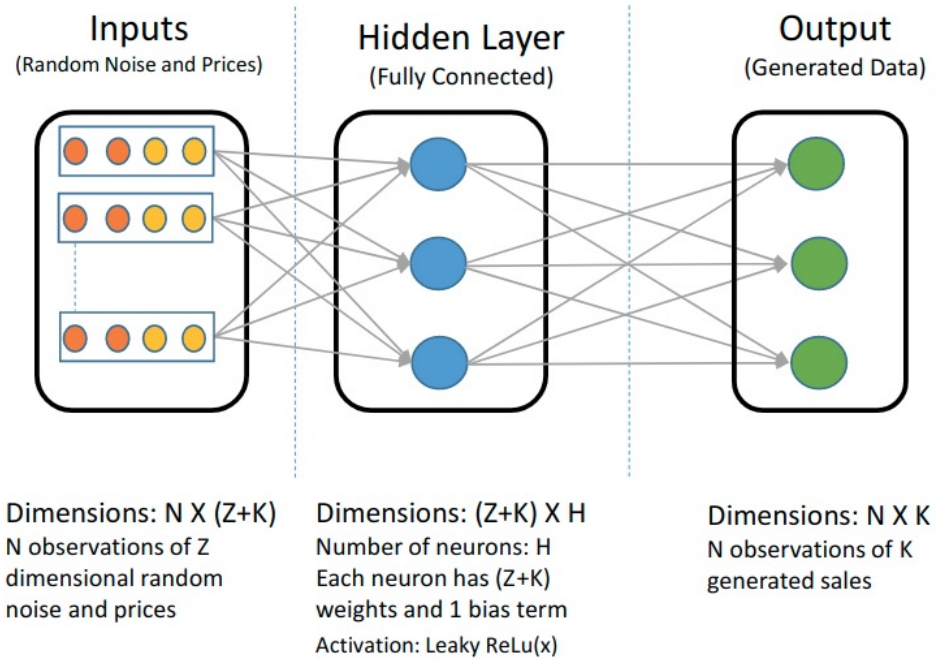
This table shows the summary statistics for the Monte Carlo data generated using the process described in Section 3.3. The data is for 200 customers for 52 weeks across 5 brands. Variable Visibility column lists whether a variable is public data or private data. The following variables are public for each brand: Display, Feature, Price, i.e. observed by both researcher, and the Sales variable for each brand is private data, i.e. visible only to the firm. Note that the generator in the GAN model has access to only the public data, and the private data never leaves the walls of the firm.

Variable	Mean	Std. Dev.	Min	Max	Variable Visibility
Display (Brand 1)	0.25	0.43	0	1	Public
Display (Brand 2)	0.15	0.36	0	1	Public
Display (Brand 3)	0.05	0.21	0	1	Public
Display (Brand 4)	0.16	0.36	0	1	Public
Display (Brand 5)	0.10	0.30	0	1	Public
Feature (Brand 1)	0.30	0.46	0	1	Public
Feature (Brand 2)	0.20	0.40	0	1	Public
Feature (Brand 3)	0.15	0.36	0	1	Public
Feature (Brand 4)	0.25	0.43	0	1	Public
Feature (Brand 5)	0.10	0.30	0	1	Public
log (Price Brand 1)	1.08	0.17	0.13	1.60	Public
log (Price Brand 2)	1.59	0.21	0.07	2.20	Public
log (Price Brand 3)	2.07	0.13	1.42	2.47	Public
log (Price Brand 4)	2.48	0.08	2.14	2.75	Public
log (Price Brand 5)	2.30	0.10	1.80	2.62	Public
log (Sales Brand 1)	-0.67	1.40	-5.67	5.14	Private
log (Sales Brand 2)	-2.76	1.31	-7.24	4.11	Private
log (Sales Brand 3)	-3.92	1.47	-7.98	5.57	Private
log (Sales Brand 4)	-4.65	1.48	-9.44	1.88	Private
log (Sales Brand 5)	-2.46	1.31	-6.79	4.69	Private

A.5 Design of Neural Networks in GANs

We discuss the design of the generator and discriminator neural networks. We base the design of both generator and discriminator on the original conditional GAN from Mirza and Osindero (2014). These are neural networks with only "fully-connected" hidden layers, and are represented in Figures A.3 and A.4.

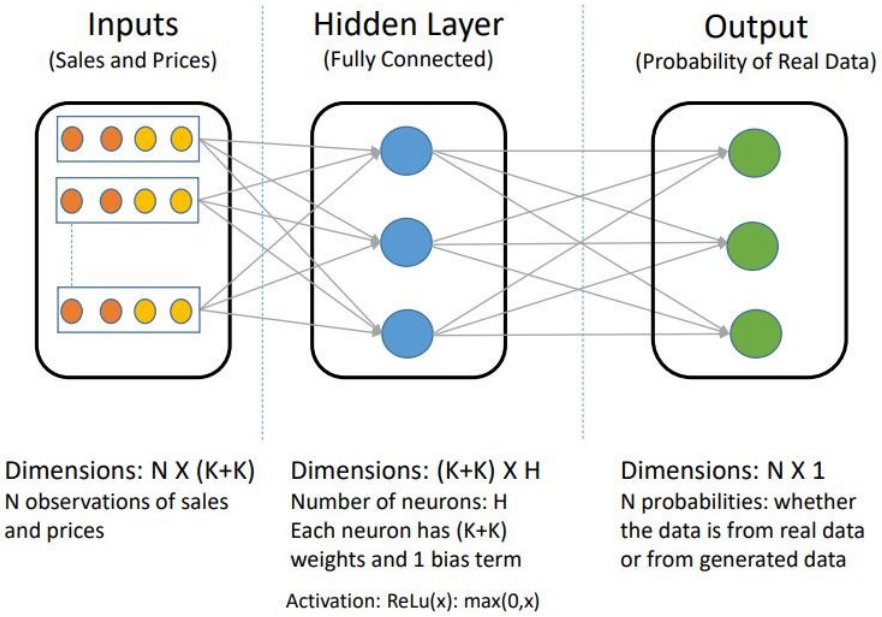
Figure A.3: Design of Generator Neural Network



As proof of concept, we employ only one hidden layer for both neural networks. We use ReLU activation in the discriminator and Leaky ReLU activation in the generator. We also use batch normalization in the generator.[6] From previous studies, we understand that neural networks perform well in classification and data generation tasks due to the universal approximation theorem (Hornik et al., 1989; Cybenko, 1989)): Given enough neurons in the hidden layers, a neural network can approximate any function. Essentially, this property allows the generator to represent the true DGP of original unprotected data in a semi-parametric fashion: such that the more neurons included in each hidden layer, the better able the network is to capture the true DGP. As the number of parameters grows, however, the model increases an estimation "burden" on the data as well, which can potentially yield a nonlinear relationship between the number of neurons and accuracy of the generator. We thus explore the efficacy of hyperparameters, such as the number of neurons, in influencing performance on accuracy and privacy metrics in the Online Appendix A.7.

[6] We discuss the data flows in GANs and robustness to different network architectures in the Online Appendix A.6 and A.8 respectively.

Figure A.4: Design of Discriminator Neural Network

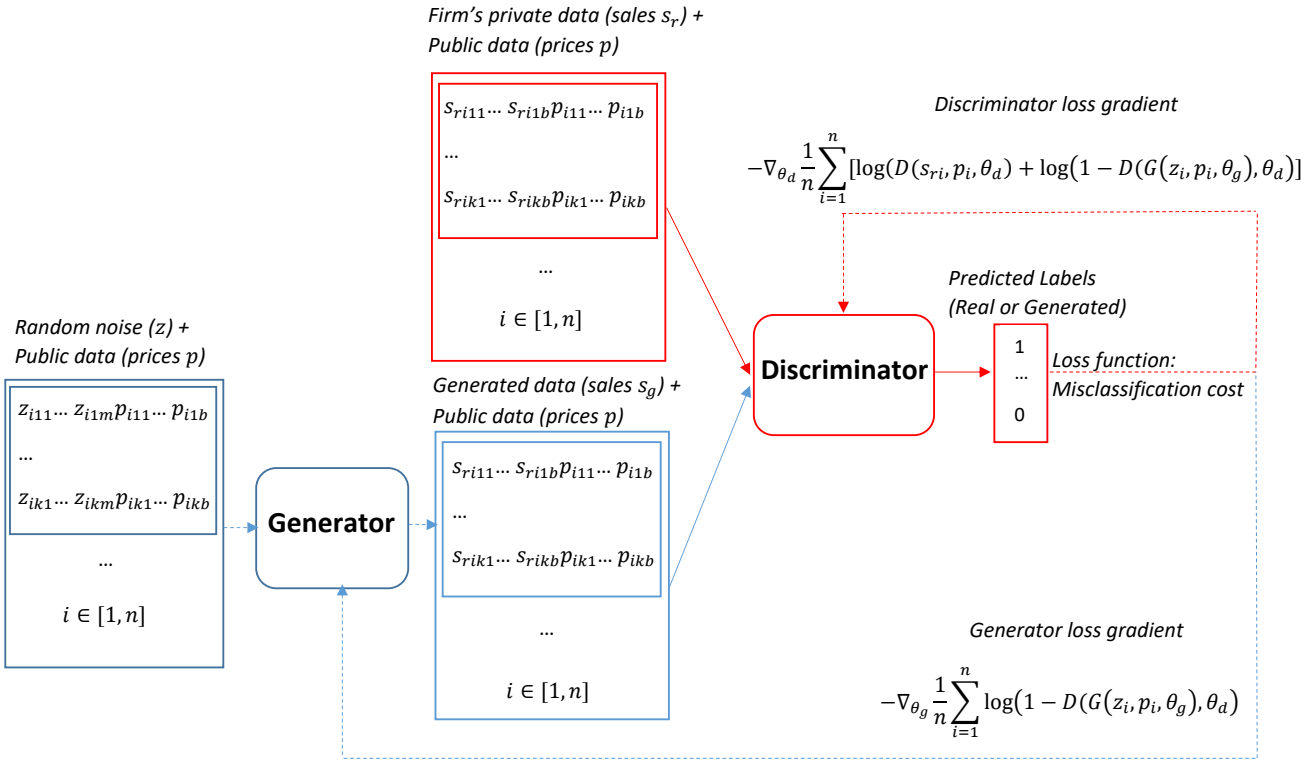


A.6 GAN Training

Figure A.5 shows the training process with gradients flow used to update the parameters. For each iteration, the generator samples a mini-batch size of n, i.e., n image equivalents of customer data. Each image equivalent data i in the mini-batch consists of a $k \times m$ random noise, where k is the number of time periods and m is the dimension of the vector sampled for each time period. It also observes the publicly available prices across brands: p_{iKB} for a brand B at time period K. The generator then outputs "generated" data s_g for n customers. The discriminator, having access to the data provider's private data, is provided samples n customers sales data: s_r. Thus, the discriminator receives a mini-batch samples for the "real" data and the "generated" data. The discriminator as a binary classifier then predicts labels for the "real" and "generated" data, and the misclassfication loss gradients are used to update the parameters: $J_d(\theta_d)$ for the discriminator, and $J_g(\theta_g)$ for the generator.

Figure A.5: GAN Training with Loss Function Gradients

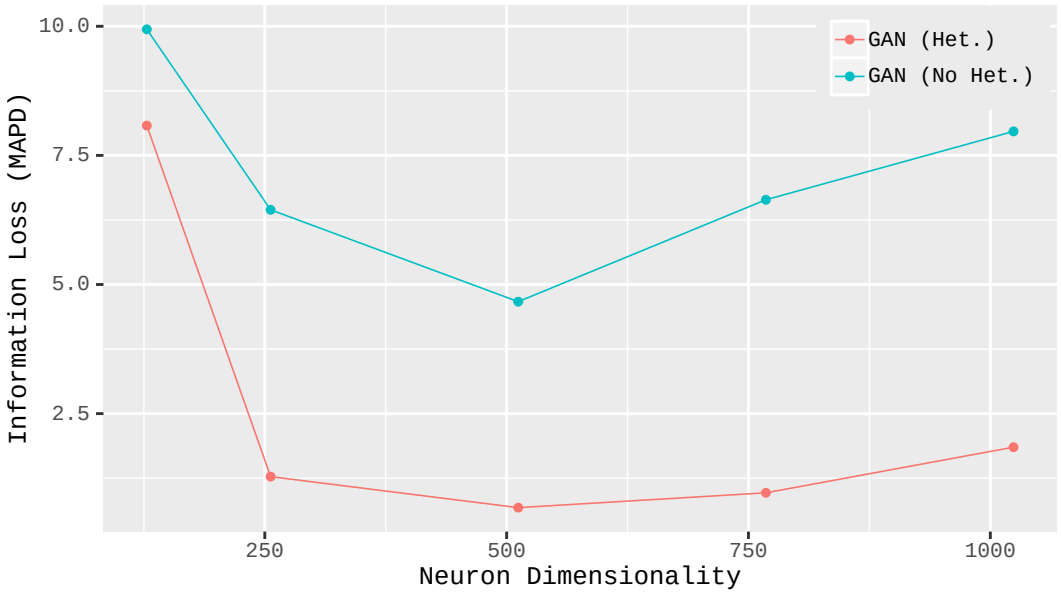
This figure shows the each iteration of the training process for the conditional GAN with a training batch of n customers who chose from b brands. Note that the actual, private data s_r is only accessible by the firm's private discriminator. z_{ikm} is the m-dimensional random noise draw for customer i at time period k, p_{ikb} is the price for brand p at time period k that is observed by customer i, s_{rikb} and s_{girb} are the "real" sales and "generated" sales respectively for a customer i at time period k for brand b. The data and gradients flow in highlighted in red are private, i.e. visible only to the data providing firm, and those in blue are public and visible to the researcher.



A.7 Relationship between Hyperparameters and Accuracy

We consider in this section the relationship between number of neurons and accuracy of the GAN's replication of the original data distribution for the Monte Carlo data.

Figure A.6: Information Loss vs. Neuron Dimensionality.



The values reported in Figure A.6 are the minimum MAPD values across fifty seeds for each of the generator types - GAN (Het.) and GAN (No Het.), and also number of neurons.[7] We find that the best performance is for the GAN with heterogeneity and 512 neurons, as it has an information loss, MAPD, of 1.2, or total of 1.2 percent difference in the regression coefficients from the true data and GAN generated data. We also find a decreasing relationship between number of neurons and information loss.

In order to test if the MAPD values differ across the generators, GAN (Het.) and GAN (No Het.), and for different values of number of neurons, we conduct two-tailed t-tests for the difference. We first conduct a cross generator-type t-test for each of the number of neurons. We find that for all values of number of neurons, we reject the null that GAN (Het.) and GAN (No Het.) have the same MAPD ($p < 0.01$). We next consider whether for a given generator type, GAN (Het.) or GAN (No Het.), if the MAPDs' differ statistically for the different values of number of neurons. We find that for both GAN (Het.) and GAN (No Het.), the MAPD is statistically different for number of neurons 128, 256, and 512. However, the differences in MAPD after this point are not significantly different from those of 512. This finding indicates diminishing returns in accuracy for increasing GAN complexity. Furthermore, we also find that generator trained to incorporate heterogeneity, GAN (Het.), has consistently lower information loss than the generator trained without heterogeneity - GAN (No Het.).

[7] We follow a common convention from computer science literature where the neuron sizes are divisible by multiples of 8, such as in increments of 128 or 256.

A.8 Model Architecture and Accuracy

In this section, we evaluate the accuracy of GANs with different model architectures. We use GANs with the following model architecture as the baseline model:

1. Generator model:
   (a) Activation function: Leaky ReLU
   (b) Batch normalization
   (c) Random noise distribution: Uniform
   (d) Number of neurons: 512
2. Discriminator model:
   (a) Activation function: ReLU
   (b) Number of neurons: 512
3. Training procedure: With Heterogeneity

For the activation functions for both the generator model and discriminator model in the GAN, we consider ReLU and Leaky ReLU activation functions. The rectified linear unit activation function, also commonly known as ReLU activation, is defined as:

$$ReLU(x) = max(0, x)$$

Thus, ReLU introduces a non-linearity at $x = 0$. The leaky rectified linear unit activation function, also commonly known as Leaky ReLU activation, is defined as:

$$LeakyReLU(x) = max(x, 0.2 \times x)$$

The Leaky ReLU activation potentially helps the sparse gradients problem of ReLU, as the output of the activation is not set to zero when x is negative. Thus, we explore both Leaky ReLU and ReLU activations as the activation functions. We also introduce batch normalization in the intermediate layer of the generator model, as batch normalization can potentially help alleviate mode collapse problems while training (Xiang and Li, 2017). Finally, we explore both uniform noise and Gaussian noise for the random noise that is used by the generator model to generate samples. The uniform noise is drawn randomly between -1 and 1, whereas the Gaussian noise is drawn from a standard normal distribution with mean 0 and standard deviation 1.

Table A.10 shows the MAPD for different activation functions, noise distributions and batch normalization. We find that compared to baseline, MAPD measures for the other model variants are not substantially different, as the MAPD ranges from 0.0056 to 0.0215, with the baseline model MAPD of 0.011.

Table A.10: GAN Accuracy with Different Architectures

Model	Generator Activation	Discriminator Activation	Noise Distribution	Batch Normalization	MAPD
Baseline	Leaky ReLU	ReLU	Uniform	Yes	0.0111
2	ReLU	Leaky ReLU	Uniform	Yes	0.0076
3	Leaky ReLU	Leaky ReLU	Uniform	Yes	0.0056
4	ReLU	ReLU	Uniform	Yes	0.0215
5	Leaky ReLU	ReLU	Uniform	No	0.0094
6	Leaky ReLU	ReLU	Gaussian	Yes	0.0106

References

Christen, Markus, Sachin Gupta, John C Porter, Richard Staelin, Dick R Wittink. 1997. Using market-level data to understand promotion effects in a nonlinear model. *Journal of Marketing Research* 322–334.

Cybenko, George. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* **2**(4) 303–314.

Hess, Sibylle, Wouter Duivesteijn, Decebal Mocanu. 2020. Softmax-based classification is k-means clustering: Formal proof, consequences for adversarial attacks, and improvement through centroid based tailoring. *arXiv preprint arXiv:2001.01987* .

Hornik, Kurt, Maxwell Stinchcombe, Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* **2**(5) 359–366.

Leeflang, Peter SH, Dick R Wittink. 2000. Building models for marketing decisions:: Past, present and future. *International journal of research in marketing* **17**(2-3) 105–126.

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

Park, Chang Hee, Young-Hoon Park. 2016. Investigating purchase conversion by uncovering online visit patterns. *Marketing Science* **35**(6) 894–914.

Schneider, Matthew J, Sharan Jagpal, Sachin Gupta, Shaobo Li, Yan Yu. 2018. A flexible method for protecting marketing data: An application to point-of-sale data. *Marketing Science* .

Wittink, Dick R, Michael J Addona, William J Hawkes, John C Porter. 1988. Scan* pro: The estimation, validation and use of promotional effects based on scanner data. *Internal Paper, Cornell University* .

Xiang, Sitao, Hao Li. 2017. On the effects of batch and weight normalization in generative adversarial networks. *arXiv preprint arXiv:1704.03971* .