A Deep Generative Model for Missing Data Imputation

by

Rozhina Ghanavi

A thesis submitted in conformity with the requirements
for the degree of Masters of Applied Sciences
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

A Deep Generative Model for Missing Data Imputation

Rozhina Ghanavi

Masters of Applied Sciences

Graduate Department of Electrical and Computer Engineering

University of Toronto

2021

Machine learning relies on data. However, real-world datasets are far from perfect. One of the biggest challenges in working with these datasets is missing data. In this work, we present a novel deep generative model for missing data imputation. What makes our method unique is the focus it puts on classification accuracy while it imputes missing data. This makes our model particularly useful for classification problems. We formulate our proposal as a sequential game and show that it learns the true data distribution. Furthermore, we propose a new algorithm for optimizing our objective. Our proposal is able to learn the feature importance and impute critical features more accurately. Experimental results show our method outperforms existing methods in terms of classification accuracy.

Acknowledgements

I would like to pay my special regards to my thesis supervisor, Ben Liang. I was privileged to be advised and guided by Ben and to do the research I truly enjoy under their supervision. I would like to recognize the valuable comments of Ali Tizghadam in shaping this work. I wish to express my deepest appreciation to my committee members, Baochun Li and Shahrokh Valaee, for their support.

I want to thank the WCL group members at the University of Toronto, especially Erfan Meskar, for their collaboration.

On a personal note, I wish to show my gratitude to my undergraduate supervisors, Maryam Sabbaghian and Halim Yanikomeroglu, who did not stop supporting me to this day. I want to thank my friends for their company during every peak and valley Pegah, Saman, and Samin. I am grateful to Violet and Mehran for making Toronto home for me. Lastly, I am forever appreciative of my Mom for being my role model, my Dad for his science chats, and my brother for bringing joy to my life.

Contents

List of Tables

List of Figures

Chapter 1

Background

1.1 Introduction

A correct prediction, estimation, and inference model rely on a good dataset. Missing data is an inevitable issue in many applications due to faulty data collection, costly or dangerous measurement, and many other reasons. Some examples of missing data are a sensor might fail to capture part of the network's information. A camera may be partially blocked and cannot capture the entire picture. It can be hard to run some medical tests on all the patients and users may refuse to answer every question in a survey.

Scholars have been studying this problem for decades, and there is a rich library of relevant literature to draw on. A survey of missingness models, challenges, and some recent works can be found in [1]. Some examples of fields where instances of missing data can occur are economics [2], computational biology [3], and psychology [4]. Many published works in the field of machine learning and statistics covered missing data problems, including [5, 6]. In [7], the authors brought to light the problem of missing values in network traffic classification. They concluded that this issue is common in network traffic flow datasets. For this application, the deletion of all the data entries with at least one feature missing is not an option since it will delete a considerable

portion of the datasets.

Despite the large body of work produced in this field, missing data imputation is still an open problem. Previous methods such as deletion, mean imputation, and interpolation are insufficient to handle problems caused by missing data. Deep generative imputation methods have attracted much attention in recent years [8–10]. The main benefit of using generative models is that they make the uncertainty estimation of imputed value possible with multiple imputation [11].

In this thesis, motivated by the major problem caused by missing data in network traffic classification, we develop a generative imputation model to impute missing data. Our proposal is general and can be applied to any tabular dataset. The significant difference between our method and other work in this field is that we take into account the classification accuracy as our primary motivation. This work makes two principal contributions. First, it proposes a new generative model for imputing missing data. Second, with experimental results, we show the effectiveness of the proposed method regarding the final classification accuracy compared to other existing techniques.

1.2 Overview of Current Literature

In general, works in the area of missing data can be categorized by their application areas. This includes computational biology, survey analysis, transportation systems, image and video processing, machine learning, and statistics. The rest of this section summarizes these works.

1.2.1 Computational Biology

In [3], a comparison between a singular value decomposition (SVD) based method, a K-nearest neighbours (KNN) based approach, and an average row estimate of missing values is presented. They sort these methods based on their robustness. According to

their work, the KNN based approach is more robust than the SVD based method, and the SVD based system is superior to the row average method. In another work, [12], a way to efficiently estimate missing data value is developed. To this end, the author proposes a method called local least squares imputation. The authors of [13] provide a survey on multiple imputation methods and their sensitivity analysis in medical research. In [14], the authors perform principal component analysis (PCA) over incomplete dataset from plant traits.

1.2.2 Social Sciences

Missing data often occurs in survey data analysis in many fields, specifically in social sciences. For example, individuals do not always answer all the questions in a survey, some will not continue participating in the follow-up research leading to incomplete data, etc. In finance, [15] proposes some adjustment on the missing data when consumers do not answer all the questions in a survey. In psychology, [4] gives a description of how much effect missing data in the psychological forms can have on their analysis. They also provide some insights into applying existing imputation methods in other literature for their surveys. In [16] the authors compare some of the proposed approaches for handling missing data in a psychological study with missing entries.

1.2.3 Transport Systems

It is common to have missing data in transportation traffic datasets due to the failure of sensors. In [17], probabilistic PCA is used to carry out online data imputation before further analysis. In [18], the authors propose an approach for missing data imputation based on probabilistic PCA. A study on the effect of missing data on neural network performance for forecasting transportation traffic is presented in [19]. The authors of [20] propose a generative adversarial networks (GAN) based approach dealing with missing values for generating synthetic airline passenger name records.

1.2.4 Image and Video Processing

Missing data cause many image and video processing problems, especially for processing old videos. In [21], the authors presented a Bayesian approach for finding missing data treatment alongside motion estimation for video analysis. In [22], an approach for image denoising and blind inpainting is suggested. A substitution approach for both texture and structure in the missing region of images is proposed in [23]. In [24], a normalized radial basis function neural network is presented as a way to deal with missing data and noisy information in visual processing. In [5], the authors present how to calculate the posterior distribution latent variables in a variational autoencoders (VAEs) architecture in the presence of missing data. This work gives an exact solution and multiple approximations of the precise solution of this problem. In [25], a feedforward network is trained under the existence of missing data. In [6], the authors discuss unsupervised learning under the existence of missing data. They develop learning and inference methods and provide empirical results for their suggestions.

1.2.5 Netflix Competition

Some work in the area of missing data is based on the Netflix challenge. In [26], a large scale matrix completion is carried out with the use of low-rank solutions. In [27], the authors present a probabilistic version of PCA, which can handle missing data.

1.2.6 Graph Analysis

Missing data imputation has been considered in the analysis of networks as graphs. Examples of these graphs are protein-to-protein interaction networks and seed exchange in the farming. In a experiment-based paper, [28], the authors propose a variational EM algorithm for the inference of stochastic block models under the assumption of missing data.

1.2.7 Network Traffic Classification

In [7], the authors brought attention to the problem of missing values in network traffic classification. For handling this issue, they propose a method of feature transformation. However, with the newly developed deep generative methods, there are better ways to solve this issue and obtain better results.

1.2.8 General Methods for Tabular Dataset

In this section, we summarize works that are more general and tested for multiple applications with tabular datasets. This category consists of two different lines of works.

The first group studies *existing* imputation methods on multiple datasets. This group's ultimate goal is to either conclude which approach is more suitable for certain conditions or build software for general users. Examples of these works include [9, 10, 29–31]. The authors in [9] give a comparison between existing works in the area of missing data imputation and deep generative models. The same work then proposes enhancement for some of these approaches. In [29], the authors compare four different methods for missing data imputation and run extensive experiments to compare them. Softwares for automatizing missing data imputation are proposed in [10, 30, 31].

The second group presents *new* data imputation techniques and tests them on different datasets. In [32], a density-based approach for handling missing data in datasets is proposed. In [33], the authors present a random forest-based approach to impute different kinds of data. The authors of [34] propose a Markov chain Monte Carlo (MCMC) approach for image inpainting. The most recent works in these categories are based on deep generative models. In [8], the authors propose a new framework for designing variational autoencoders (VAE) in order to estimate missing data correctly. This method outperforms supervised methods when it is applied to incomplete data. A denoising autoencoder based imputation model is proposed in [35]. In [36], a novel method for data imputation based on GAN is presented and is proven to work more efficiently than other

data imputation methods. More details about these methods are given in Section 1.4.2.

1.2.9 General Methods for Time Series

Besides tabular data, another important data type is time series. Data in many applications in health care and finance depend on time, and their structure changes depending on when the data is sampled. These data are in the format of time series, and the existence of missing data in multivariate time series makes working with them challenging. In [37], an imputation method based on GANs is presented for imputing lost elements in time series. In [38], authors present a VAE based method for missing data imputation and dimensionality reduction in time series. An imputation approach based on adversarial training for long-range sequences is proposed in [39].

1.3 Categorization of Prior Works

In this section, we further categorize the aforementioned works in Section 1.2.

1.3.1 Summary of Research Methods

Prior works may propose new methods or study existing methods. For performance evaluation, they may provide analytical results, run experiments, or conduct experiments with real-world applications.

	New methods	Existing methods
Analysis based	[5, 8, 26, 36, 40]	[2, 28]
Experiment based	[6, 9, 22–25, 27, 32–35, 37–39, 41, 42]	[3, 4, 10, 12–21, 29–31, 43]

Table 1.1: Summarizes of the research methods of the prior works.

Table 1.1 presents the four different cases of research methods in missing data impu-

tation. The table includes four different categories which we briefly mention what they mean. New and analytical works include works that proposed new methods for handling missing data and analytically proved their effectiveness. Analysis based existing works part of the table includes works that provide analytical results over existing methods in the literature. New and experiment-based works cover works that proposed new methods for handling missing data and conclude their effectiveness by experimental results. Lastly, applications and comparison of existing methods is the category that includes work that uses and compares different existing methods in the literature of missing data in different applications.

1.3.2 Models of Missingness

One crucial concept which groups different works in this field is their choice of missingness model. Missingness, in this setting, refers to the condition in which the data is missing. In [43], the authors define three classes of missingness models based on the way the data are missing. In order to understand the missingness models as defined in [43], we fist present some important terms. Consider a dataset of n samples, each with d features. Let x be the $n \times d$ matrix representing the feature set, t be the n-dimensional vector of labels, and ω be the missingness matrix, be an indicator showing whether a feature is missing. We define the variable q is as $q = (x, t)$. We write $q = (q_{vu})$, where $v = 1, ..., n$ are data rows and $u = 1, ..., d$ are features. We assume it is possible to separate the observed and missing entries as $q = (q^{\text{obs}}, q^{\text{mis}})$, and $\omega = (\omega_{vu})$, where $\omega_{vu} = 1$ if q_{vu} is observed and $\omega_{vu} = 0$ otherwise.

The following categorization of missing data is presented in [43]. The first category is referred to as missing completely at random (MCAR). Data is MCAR when ω does not depend on q. In this case $p(\omega|q) = p(\omega)$. The second category is missing at random (MAR). This model states that ω does not depend on q^{mis}. This means that $p(\omega|q) = p(\omega|q^{\text{obs}})$. The final category is not missing at random (NMAR). In this category, the

missingness ω depends on both the observed and unobserved value.

Works summarized in Section 1.2 can be categorized by the model of missingness as follows.

- MAR models only: [3, 5, 8, 9, 12, 13, 21–25, 27, 34–42].
- NMAR models only: [6, 10, 30, 31]
- All MCAR, MAR, and NMAR models: [28, 29, 43].
- A special combination of NMAR and MCAR: [14, 33].

The proposed method in this thesis can be applied in any type of missing data.

1.4 Methods for Missing Data Imputation

Missing data has been a challenge in many statistical studies. Many suggestions for how to overcome this issue can be found in the relevant literature. These approaches can be categorized into two major groups:

- Naïve approaches
- Machine learning approaches

A summary of the approaches in these categories is given in Figure 1.1. In the rest of this section, we first summarize naïve approaches. Then, we discuss machine learning approaches in detail because they are the main focus of this thesis.

1.4.1 Naïve Approaches

Naïve approaches in handling missing data are often easy to implement approaches that are wasteful of data. The advantage of these approaches is their generality, meaning they can be applied to almost any dataset with any characteristics. On the other hand,

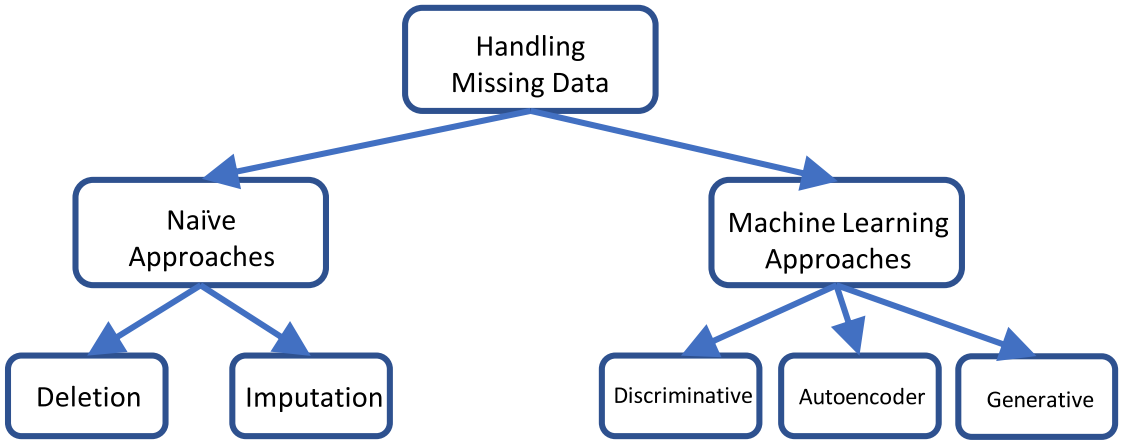

Figure 1.1: Approaches to handle missing data.

their disadvantages are wasting useful information and potentially causing harm to the inference model. Naïve approaches for handling missing data fall into the two general formats: deletion and imputation. In the rest of this section, we will name examples of these groups.

Deletion

Deletion means deleting part of the data, which includes some form of missingness. Examples of handling missing values with deletion include the following.

- List-wise deletion: this method chooses to delete any data entry that contains at least one missing feature. This method is easy to implement because it does not require any computation. However, it will remove a considerable amount of data.

- Variable deletion: in this case, an entire feature column is dropped if one entry is missing. This approach is especially problematic in small domain datasets.

- Pair-wise deletion: in this method, data entries are dropped after some analysis. These analyses may vary depending on the application and the dataset. An example is when one cannot tolerate one particular missing feature; in this case, they may

choose to drop any cases with that particular feature being missed.

Imputation

- Mean imputation: this means taking the mean of an entire feature column and replacing any missing point in that column with this value.
- Case mean imputation: this is similar to the previous approach but, instead of an entire feature column, the mean is taken for individual groups. This could be different classes or even different groups based on other values.
- Zero imputation: in this approach, any missing value is replaced with zero.
- Interpolation: this approach means fitting an easy model to observed data points and filling the missing values with this model.
- Last observed carried forward
- Worst/best case imputation: this method is popular for categorical values, where either worst or best value is chosen for the imputation.

As mentioned before, these approaches are easy to implement on any dataset. However, as our datasets become more challenging the need for a good imputation model grows.

1.4.2 Machine Learning Methods for Missing Data Imputation

In this section, we discuss existing imputation approaches that use machine learning since they are the focus of this thesis.

Discriminative Approaches

In [33] the authors focus on finding a discriminative approach that can handle missing features in all types of data ,continuous or categorical, simultaneously. To this end, they

proposed an iterative method based on a random forest model.

In [30, 31], an imputation method based on the chained equation is presented. This approach, multivariate imputation by chained equations (MICE), is a multiple imputation approach. The idea is to use the best imputation approach, chosen from a candidate set, for individual columns (features). They present software that chooses the best imputation method for each column individually. Each column is imputed with its own best method. Then, all the columns are pooled together to form the completed dataset. The algorithms to choose from are, mean matching, Bayesian linear regression, linear regression, logistic regression, linear models, multinomial logit model, ordered logit model, linear discriminant analysis, and random sampling from the observed data. For each feature's imputation, one of these methods is chosen, and the column is imputed accordingly. MICE is a used software for imputation. Hence, it is one of our benchmarks.

Autoencoder-based Approaches

An imputation method proposed by [8] is based on VAE. In [34], the author use a complete dataset to learn a sampling method, and then they perform image inpainting using learned latent space.

In [40] the authors present an approach based on importance-weighted autoencoder to learn deep latent variables in the presence of missing data.

GAN-based Approaches

This section gives an overview of more recent works in this field. These works use generative models to handle missing data. In generative adversarial imputation networks (GAIN) [36], the authors propose a model based on the generative adversarial networks (GANs) [44] for imputation of missing data entries in tabular datasets. To understand GAIN [36], we first briefly overview how GAN works. GAN consists of a generator and a discriminator. The generator is a multilayer perceptron generating fake samples, and

the discriminator is another multilayer perceptron trying to maximize the probability of assigning the right label to train (real) data or generated (fake) data. GAN aims to teach the generator to produce real looking samples by modeling the optimization problem as a two-player minimax game between the generator and the discriminator. Borrowing GAN's idea, GAIN imputes missing data by generating the missing parts in data entries, where the problem is formulated as a two-player game between the generator and discriminator. The generator generates the data features. The discriminator checks every single feature for whether it is imputed or observed. More details about this algorithm are presented in Chapter 3. In [42] the authors further propose a data prepossessing method based on [36] for missing data imputation and handling imbalanced datasets.

1.5 Contribution and Outline

In this work, we study classification under the existence of missing data. This work is motivated by the challenges caused by missing data for classification in general, and specifically, in network traffic classification [7]. To this end, we propose a new imputation method based on GAN and GAIN. Our imputation method is called generative adversarial imputation networks (GACN). It is a deep generative model that learns to impute missing data while taking the classification accuracy into account. In GACN, a network helps the imputation method to diagnose which features are more important in terms of accuracy. Those features are then imputed more accurately. To the best of our knowledge, there is no prior work considering the information from the classification accuracy to better impute missing data.

We first present the structure of GACN, which consists of three deep neural networks. Then we discuss the game theoretic model in this setting. After that, we analytically show that there exists an optimal point in the proposed game setting, which returns the

true data distribution. This point is where the players learn to impute missing features with the actual values. We define a weighted loss function based on the three networks in our model to optimize our objective. We then present an iterative three-step optimization approach to solve the problem. We further present a new method, which is termed semi-supervised (SS)-GACN, which does not need to have all the labels in the dataset and can work with partially labeled data. This variant of GACN makes it more general.

For testing our method, we use an existing network flow dataset. Our experiments and simulation show the effectiveness of GACN in comparison with different methods. We have chosen GAIN, which is the state-of-the-art imputation method, as one of our main benchmarking algorithms. We have also compared our results with mean imputation, as a basic method, and MICE, as a commonly used approach. We conclude the following key points:

- GACN outperforms GAIN, MICE, and mean imputation in terms of reaching higher accuracy faster.
- As the missingness rate increases, GACN becomes more promising than GAIN in keeping reasonable accuracy.
- The average RMSE of feature imputation for both GACN and GAIN is similar. However, individual feature-based RMSE is different in GACN and GAIN. GACN imputes more important features better.
- In the case of partially labeled data, the performance of GACN drops while SS-GACN maintains excellent performance.
- By studying our method on other existing datasets, Spam and Credit, we first show GACN is a general method. Second, we conclude that certain characteristics of the network flow dataset makes GACN performs substantially better than other methods on this dataset.

The rest of this thesis is organized as follows. Chapter 2 presents our system model, the framework we choose for classification, and how we defined a real-world dataset from network flows. In Chapter 3, we present our algorithm for missing data imputation. The approach we propose is aiming for better classification results after imputation is completed. The same Chapter includes analytical results for showing why our imputation framework is useful for missing data imputation. Chapter 4 presents a variation of our proposed method, which enables us to work with a partially labeled dataset as well. Chapter 5 is devoted to the experimental results. Finally, Chapter 6 concludes this thesis.

Chapter 2

System Model

2.1 Classification

Our goal is to maximize classification accuracy, given any real-world dataset which consists of missing values. To this end, we define our problem as follows. Suppose data vector, $X = (X_1, X_2, \ldots, X_d)$, is a random vector in $\mathbb{R}^d$, and x is a realization of X.[1] Every X is labeled with a target value, T, showing the class to which it belongs, and t is a realization of T. Let $M = (M_1, M_2, \ldots, M_d)$ in $\{0, 1\}^d$ be a random vector, which we call the *missingness vector*, so that if a feature value X_i is missing, $M_i = 0$. Our goal is to impute unobserved entries of x, with an imputation algorithm in a way that maximizes the classification accuracy. The imputation algorithm should be such that it outputs x_i if and only if $m_i = 1$. Hence, when an item is observed, it forces the algorithm to output the same observed value. While if an item is unobserved, i.e., $m_i = 0$, the algorithm generates a value for the missing component and the output of the algorithm is the imputed value.

We consider the training dataset, $\mathcal{D} = \{(x^1, t^1), (x^2, t^2), \ldots, (x^N, t^N)\}$, with each sample drawn independently from the joint distribution of X and T. For the classification

[1] In general, we use uppercase letters to represent random variables and lowercase letters to represent the realization of random variables.

accuracy metric, we choose the cross-entropy loss, which is defined as follows.

$$\sum_{k=1}^{N}\sum_{\iota=1}^{I} t_{\iota}^{k}\log(C(\hat{x}^{k}))_{\iota}, \tag{2.1}$$

where $\hat{x}^k$ denotes the imputed value corresponding to x^k, C can be any desired classifier assigning how likely it is for any given $\hat{x}^k$ to belong to either of the ι classes and I is the total number of classes. It is noteworthy that any general metric can be used instead of cross-entropy. If another metric is chosen, then the proposed algorithm in Chapter 3 needs to be modified slightly concerning the new metric.

2.2 Network Flows

This thesis uses a network traffic flow dataset where each flow is labeled with its actual application. The objective is to identify whether a flow is delay-sensitive or delay-tolerant. In other words, the goal is to classify data entries based on their quality-of-service (QoS). As a result, there is a need to find a mapping between the applications and the QoS labels. The real-world example which we consider is a combination of ISCX VPN-nonVPN [45,46] and ISCX Tor-nonTor [47, 48] datasets. These datasets are PCAP files from encrypted TCP flows. Borrowing the idea in [49] we extracted 266 features and 22 applications from the dataset. Some examples of the 266 extracted features are the following:

- Mean and variance of packet length
- Mean and variance of the payload length
- Source and destination port
- Min and max window size
- Throughput
- Mean and variance of the frame rate

- Mean, variance, and FFT of interval time

We first define eight general application classes and map the 22 applications to those eight classes for labelling applications with their representative QoS target. Then we map each of these eight classes to the delay-sensitive group or the delay-tolerant group. Figure 2.1 shows the applications which are delay tolerant, and Figure 2.2 presents the delay-sensitive applications.

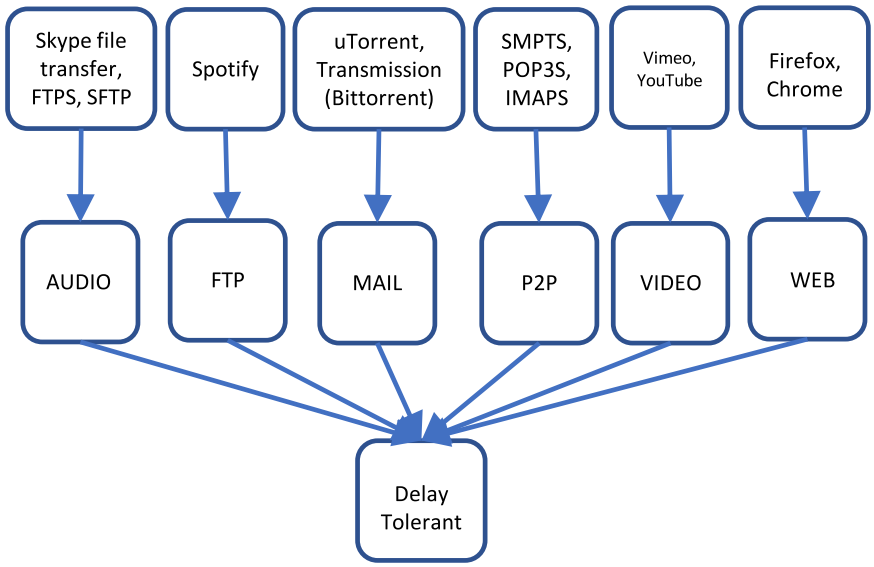

Figure 2.1: Delay tolerant applications.

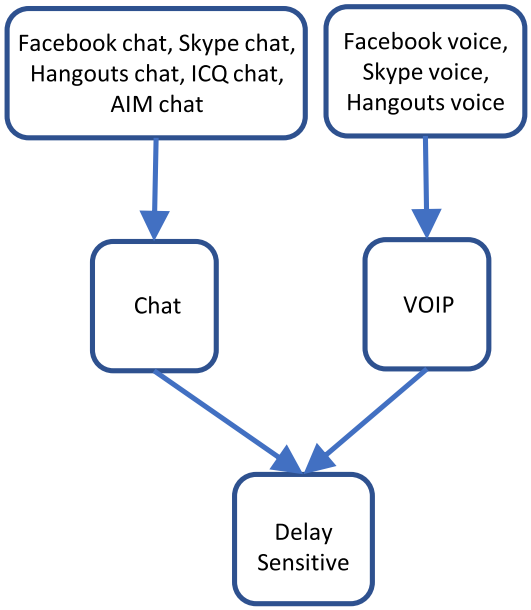

Figure 2.2: Delay sensitive applications.

Chapter 3

Generative Adversarial Classification Network

3.1 GACN Framework

This thesis's main objective is to minimize the cross-entropy loss in (2.1) for any tabular dataset under the presence of missing data. For this general case, optimization implies the need for missing data imputation. Generative adversarial classification networks (GACN) is our proposed algorithm for missing data imputation while considering the classification accuracy. This imputation procedure aims to have a perfect generator that can generate values for missing data entries close to what might have been seen in reality by learning the data distribution. GACN consists of three neural networks. Here, we consider perceptron networks, but our work is not limited to this choice. These three components are the generator network, the discriminator network, and finally, the accuracy network. We next present the details of GACN.

3.1.1 The Generator

The definition of the generator is as follows. We first define a noise vector, $Z \in [0,1]^d$. This noise vector is the input to the generator. We express the generator as $G(X, M, Z; \theta_g)$, a differentiable function taking value in $\mathbb{R}^d$, which is multilayer perceptron with parameters θ_g. It takes $M \odot X + (1 - M) \odot Z$ as input, where $\odot$ denotes element-by-element multiplication. It outputs a vector of imputed feature values:

$$\check{X} = G(X, M, Z) \tag{3.1}$$

Notably, G is a general generator that generates a value for all the feature entries, both observed and unobserved. However, if a value is observed, we want to have the observed value as the algorithm's output. That is why it is essential to update the output as

$$\hat{X} = \begin{cases} X_i & \text{if } M_i = 1, \\ \check{X}_i & \text{otherwise.} \end{cases} \tag{3.2}$$

Given the above equation, now the entries of $\hat{X}$ are equal to X for the observed values and are equal to the generated values for missing features.

3.1.2 The Discriminator

The discriminator, D, is a multilayer perceptron defined by $D(\hat{X}; \theta_d)$, with the network parameters θ_d. It outputs a vector in $[0,1]^d$ for the probabilities of the data entries in $\hat{X}$ being observed instead of imputed. We will show in Section 3.2 that for the theoretical results to hold, it is necessary to define a selection vector, $R \in \{0,1\}^d$ [36]. It is noteworthy that learning a good discriminator here is harder than in GAN. Because here, the discriminator needs to assign the probability of being imputed to every single feature, while GAN's discriminator job was to decide simply if the entire generated set is real

or fake. This is intuitively the reason behind the existence of the selection vector. It is needed to give the training of D some information, on the likelihood that a feature value is missing for any given data sample, to make this job easier. The selection vector R is given by

$$R_i = \begin{cases} 1 \text{ if } i \neq r \\ \\ 0 \text{ if } i = r, \end{cases} \tag{3.3}$$

where r is uniformly sampled from 1, ..., d. With this definition, now the discriminator needs to decide only whether the feature $i = r$ is imputed or observed, making the discriminator's job no harder than what is in GAN. Later in the proof section, we will show that there is a need for a hint vector, to be defined based on R, which contains enough information from the missingness vector so that the proposed method ends up learning the data distribution. We then define the output of the discriminator block as the following:

$$\hat{P}(X, R) = \begin{cases} D(X)_i & \text{if } R_i = 0, \\ \\ M_i & \text{otherwise.} \end{cases} \tag{3.4}$$

Note that, in the above equation, the entries of $\hat{P}(X, R)$ are equal to M_i except when $R_i = 0$.

3.1.3 The Classifier

The classification network, $A(\hat{X}; \theta_a)$, is the last multilayer perceptron of the GACN model with parameters θ_a. At this stage, we assume there is no missingness in the target values. Then A outputs the probability of assigning the data vector to a specific target label.

3.1.4 Connecting the Components

Figure 3.1 shows how these three networks interact with each other. As shown in this figure, G receives five elements as its input: the data vector X, the noise vector Z, the missingness vector M, and the functions D and A. The generator then outputs the completed vector $\hat{X}$. The completed vector is then one of the inputs to the discriminator. In addition to the completed data vector, D also receives the selection vector and outputs the modified probability vector $\hat{P}$ of each data entry being imputed. The completed vector from the generator is also the input to the classification network, A. The classification network outputs the probability of assigning each data points to different classes.

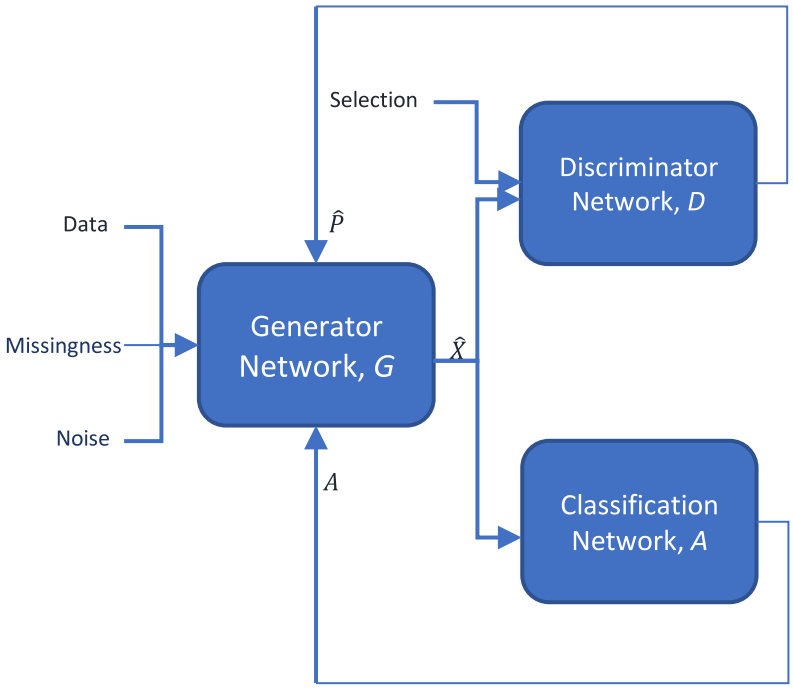

Figure 3.1: GACN model schematic.

As mentioned before, the dataset includes missing data; hence, we need to take care of the missing values for solving the softmax classification. The main idea is to build a good generator that can generate fake but real-looking values for missing data. Similar

to approaches used in [36, 44], we formed a game to train such a generator. Here, we combined three neural networks. The first one, the generator, generates replacement for the missing data. The second one, the discriminator, gives feedback about how good the generator's imputation was. The last one, the classification network, shows how much the imputation helps in terms of the final accuracy. Under this setting, a sequential game occurs between the generator and the discriminator. The generator first does the imputation, then knowing the result of this imputation, the discriminator will decide if an entry was seen or imputed. The classification network is not a player, but it actively gives feedback to help the generator and the discriminator find which feature is more critical in terms of accuracy. This feedback makes sure that for more essential features, the data imputation is performed more carefully.

For the sequential game between the discriminator and the generator, the aim is to solve the following minimax problem.

$$\min_{\theta_g} \max_{\theta_d} L(D, G), \tag{3.5}$$

where $L(D, G)$ is defined as follows:

$$L(D, G) = \mathbb{E}_{X,M,R}[M^T \log \hat{P}(\hat{X}, R) + (1 - M)^T \log(1 - \hat{P}(\hat{X}, R))]. \tag{3.6}$$

In (3.5), the multiplications are element-wise. In this equation, the discriminator network tries to maximize the probability of correctly deciding whether a feature is real or imputed. The generator, on the other hand, tries to minimize the chance of the discriminator deciding correctly. This implicitly means that the generator's goal is to impute the data so well that the imputed features are indistinguishable from the real ones.

Here, $L(D, G)$ is an expectation over three random variables' realization. The first two random variables, X and M, are the data vector and missingness vector. These two random variables are defined in Section 2.1. The last random variable R is the selection vector presented in Section 3.1.2. As mentioned earlier, this random variable's existence

is essential for assuring adequate performance from the discriminator. The dependence of (3.6) to the generator is trough $\hat{X}$. The first part of this equation, $M^T \log \hat{P}(\hat{X}, R)$, checks how well the discriminator assigns the observed values' probabilities. At the same time, the second part checks the same thing for missing and imputed values.

3.2 Analysis of Optimum

In this section, we analytically show why optimizing (3.5) is a suitable imputation technique. Similar to the proof in [36], we prove solving the above minimax problem under some conditions is equivalent to finding the data distribution. This essentially means that if we find the presented game's global optimum, then the generator knows the data distribution. Hence imputing with this generator means imputing with essentially real values. In the following, m, x, $\hat{x}$, b, and h are realizations of M, X, $\hat{X}$, B, and H respectively. As in [36], here H is a general hint vector which is provided by R.

We will show that the optimal point of the aforementioned game is where $p(x \mid h, m_i) = p(x \mid h)$ for all m, x, h.[1] Keeping in mind the fact that the optimization is carried out iteratively, we will first find the optimal distribution of the discriminator given a fixed generator. Then given the optimal discriminator, we can find the global minimum for the overall loss function.

Lemma 1. *The optimal distribution for D, given a fixed G that produces $\hat{X}$, is as follows:*

$$\begin{aligned} \hat{P}^*(\hat{x}, h)_i &= \frac{p(\hat{x}, h, M = 1)}{p(\hat{x}, h, M = 1) + p(\hat{x}, h, M = 0)} \\ &= p(M = 1 | \hat{x}, h). \end{aligned} \tag{3.7}$$

[1] We use $p(\cdot)$ to denote the probability function throughout this thesis. For discrete random variables, it returns the probability mass. For continuous random variables, it returns the probability density.

Proof.

$$
\begin{aligned}
L(D,G) &= \mathbb{E}_{\hat{X},M,H}\left[M^T \log \hat{P}(\hat{X},H) + (1-M)^T \log(1-\hat{P}(\hat{X},H))\right] \\
&= \int_{\hat{X}} \sum_{m\in\{0,1\}^d} \int_H \left(m^T \log \hat{P}(\hat{x},h) + (1-m)^T \log(1-\hat{P}(\hat{x},h))\right) p(\hat{x},m,h) dh d\hat{x} \\
&= \int_{\hat{X}} \int_H \sum_{i=1}^{d} (\log \hat{P}(\hat{x},h)_i \sum_{i:m_i=1} p(\hat{x},m_i,h) + \log\left(1-\hat{P}(\hat{x},h)_i\right) \sum_{i:m_i=0} p(\hat{x},m_i,h)) dh d\hat{x} \\
&= \int_{\hat{X}} \int_H \sum_{i=1}^{d} \left(\log \hat{P}(\hat{x},h)_i \; p\left(\hat{x},h,M_i=1\right) + \log\left(1-\hat{P}(\hat{x},h)_i\right) p\left(\hat{x},h,M_i=0\right)\right) dh d\hat{x}.
\end{aligned}
$$

Taking the derivative of the above and setting it to zero, we have,

$$
\hat{P}^*(\hat{x},h)_i = \frac{p(\hat{x},h,M_i=1)}{p(\hat{x},h,M_i=1) + p(\hat{x},h,M_i=0)}
$$

□

Now given D^* we define $\psi(G) = L(G,D^*)$. Then the minimization problem becomes the following:

$$
\min \psi(G) = \min \mathbb{E}_{\hat{X},M,H}\left(\sum_{i:M_i=1} \log p\left(M=1 \mid \hat{X},H\right) + \sum_{i:M_i=0} \log p\left(M=0 \mid \hat{X},H\right)\right). \tag{3.8}
$$

Theorem 1. *The optimal point of* (3.8) *is where*

$$
p\left(\hat{x} \mid h, m_i\right) = p(\hat{x} \mid h), \tag{3.9}
$$

almost everywhere, for all i.

Proof.

$$
\begin{aligned}
\psi(G) &= \mathbb{E}_{\hat{X},M,H}\left(\sum_{i:M_i=1}\log p\left(M_i=1 \mid \hat{X},H\right)+\sum_{i:M_i=0}\log p\left(M_i=0 \mid \hat{X},H\right)\right)\\
&= \int_{\hat{X}}\int_{H}\sum_{i=1}^{d} p\left(x,h,M_i=1\right)\log p\left(M_i=1 \mid \hat{x},h\right)\\
&\quad + p\left(\hat{x},h,M_i=0\right)\log p\left(M_i=0 \mid \hat{x},h\right)dhd\hat{x}\\
&= \sum_{i=1}^{d}\sum_{\zeta\in\{0,1\}}\int_{H_\zeta^i}\int_{\hat{X}} p\left(\hat{x},h,M_i=\zeta\right)\log p\left(M_i=\zeta \mid \hat{x},h\right)dhd\hat{x}\\
&= \sum_{i=1}^{d}\sum_{\zeta\in\{0,1\}}\int_{H_\zeta^i}\int_{\hat{X}} p\left(M_i=\zeta,h\right)p\left(\hat{x} \mid h,M_i=\zeta\right)\log\frac{p\left(\hat{x} \mid h,M_i=\zeta\right)}{p(\hat{x} \mid h)}dhd\hat{x}\\
&\quad + \sum_{i=1}^{d}\sum_{\zeta\in\{0,1\}}\int_{H_\zeta^i} p\left(M_i=\zeta,h\right)\log p\left(M_i=\zeta \mid h\right)dh\\
&= \sum_{i=1}^{d}\sum_{\zeta\in\{0,1\}}\int_{H_\zeta^i} p\left(M_i=\zeta,h\right)D_{\mathrm{KL}}\left(p\left(x \mid h,M_i=\zeta\right)\|p(x \mid h)\right)\\
&\quad + \sum_{i=1}^{d}\sum_{\zeta\in\{0,1\}}\int_{H_\zeta^i} p\left(M_i=\zeta,h\right)\log p\left(M_i=\zeta \mid h\right)dh
\end{aligned}
$$

where H_ζ^i is the space of h where $p(h \mid M_i=\zeta)>0$, and $D_{\mathrm{KL}}(q||q')$ is the Kullback-Leibler divergence and it is always non negative. In particular, $D_{\mathrm{KL}}(q||q')$ is 0 if and only if $q=q'$, almost everywhere. Hence, $\psi(G)$ is minimum if only and if (3.9) holds almost everywhere. □

Here we note the necessity of an informative H. If H does not contain enough information about M, (3.9) may not hold. Then, there is no guarantee that G learns the data distribution. In GACN as in [36] we define the hint vector that shares $d-1$ elements with M except for $R_i=0$.

Corollary 1. *If* (3.9) *holds, then,*

$$
p\left(\hat{x} \mid m_\nu\right)=p\left(\hat{x} \mid m_\mu\right)\quad \forall m_\nu,m_\mu\in\{0,1\}^d. \tag{3.10}
$$

This is true for $p(\hat{x} \mid m) = p(\hat{x} \mid \mathbf{1})$, where $\mathbf{1}$ is the all-one vector. Since $p(x|\mathbf{1})$ is the distribution of the feature values without any missing data, this concludes that at optimum the above game yields the original the data density. This concludes the proposed game will learn the data distribution.

As mentioned before, in addition to the neural networks G and D, which are the players in the previous game, we also have the classification network A. The role of A is to emphasize more critical features. Adding this term will ensure that the generator imputes with higher precision those features that are more important to accuracy. This hypothesis is validated experimentally in Chapter 4.

3.3 GACN Algorithm

In this section, we will discuss our proposed algorithm to solve the minimax problem in (3.5). In Section 3.2, we have analytically shown that there exists a global optimum for (3.5). We discussed why this global optimum is interesting for the imputation case. What is left now is to find an optimization solution for (3.5). In both the GAN and GAIN papers (i.e., [36,44]), the authors used gradient descent-ascent to optimize their minimax problem. Although this approach is popular, it has one important drawback. There is no guarantee that it will converge to an optimal point in the generator-discriminator game settings. For the GACN minimax optimization problem, we further enhance these methods, using our knowledge of classification accuracy.

Our method is based on three-step optimization. Given a fixed generator G, we update D and A. In the following, α, β, and γ are model hyperparameters, and B_D, B_A, and B_G are mini-batch sizes. Also, superscript j denotes the j-th sample in a mini-batch. For updating D, we first define the cross-entropy for a sample being observed as follows:

$$\mathcal{L}_D(m, \hat{P}(\hat{x}, h)) = \sum_{i=1}^{d} m_i \log(\hat{P}(\hat{x}, h)_i) + (1 - m_i) \log(1 - \hat{P}(\hat{x}, h)_i), \tag{3.11}$$

Then D is trained with the following objective:

$$\min_{\theta_d} \quad -\sum_{j=1}^{B_D} \mathcal{L}_D(m^j, \hat{P}(\hat{x}^j, h^j)). \tag{3.12}$$

For updating A we first define $\mathcal{L}_A$ as the cross-entropy loss for the target value t as follows:

$$\mathcal{L}_A(\hat{x}) = \sum_{i=1}^{d} \sum_{\iota=1}^{I} t_\iota^k \log(A(\hat{x}_i))_\iota. \tag{3.13}$$

Using $\mathcal{L}_A$, then we update A by optimizing with respect to the following objective:

$$\min_{\theta_a} \quad -\sum_{j=1}^{B_A} \mathcal{L}_A(\hat{x}^j). \tag{3.14}$$

Now with the given locally optimal A and D, we can update G. Based on extensive experimental results, we conclude that the best way to update the generator is to use a weighted loss function consisting of three elements, $\mathcal{L}_M$, $\mathcal{L}_G$, and $\mathcal{L}_P$, which are defined as follows.

$$\mathcal{L}_M(x, \hat{x}) = \sum_{i=1}^{d} m_i (x_i - \hat{x}_i)^2 \tag{3.15}$$

$$\mathcal{L}_G(m, \hat{P}(\hat{x}, h)) = -\sum_{i=1}^{d} (1 - m_i) \log(\hat{P}(\hat{x}, h)_i), \tag{3.16}$$

$$\mathcal{L}_P(m, \hat{x}) = -\log(A(\hat{x})) \sum_{i=1}^{d} (1 - m_i), \tag{3.17}$$

Minimizing $\mathcal{L}_M$ makes sure the generator learns to generate values close to the observed data's real values. While optimizing $\mathcal{L}_G$ and $\mathcal{L}_P$ helps better imputing missing values. $\mathcal{L}_G$ is the loss function associated with fooling the discriminator. Optimizing this loss function means the generator is so good at imputing missing values that the discriminator cannot distinguish between imputed or real features. The last loss function, $\mathcal{L}_P$, is a term associated with the classification loss. Adding this term helps the generator learn the model, which gives the highest accuracy by emphasizing the features that are

more important in terms of final accuracy.

Given these three loss functions we then update the generator as the follows:

$$\min_{\theta_g} \sum_{j=1}^{B_G} \alpha \mathcal{L}_M(x^j, \hat{x}^j) + \beta \mathcal{L}_G(m^j, \hat{P}(\hat{x}^j, h^j)) + \gamma \mathcal{L}_P(m^j, \hat{x}^j). \tag{3.18}$$

Algorithm 1 presents the pseudo-code of our algorithm.

3.4 Comparing GACN and GAIN

GACN, as illustrated in Figure 3.1, is an approach for missing data imputation while taking the classification accuracy into account. In contrast, GAIN [36] does not care about the classification accuracy and only cares about how close the imputed values are to the actual values. Thus, if GAIN is used for classification, it needs to be in a two steps approach first performing imputation and then carrying out classification step using a combination of the observed and imputed data. Figure 3.2 shows the schematic of classification when GAIN is used as the imputation approach.

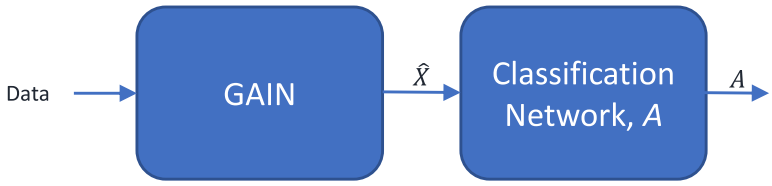

Figure 3.2: Classification using GAIN model schematic.

Algorithm 1 Pseudo-code of GACN

for a preset number of iterations **do**

(1) Discriminator optimization

Sample from the dataset $\{(x^j, m^j)\}_{j=1}^{B_D}$

Sample i.i.d., $\{z^j\}_{j=1}^{B_D}$, of Z

Sample i.i.d., $\{r^j\}_{j=1}^{B_D}$, of R

for $j = 1, \ldots, B_D$ **do**

$\check{x}^j \leftarrow G(x^j, m^j, z^j)$

for i in d **do**

if $M_i = 1$ **then**

$\hat{x}_i^j = x_i^j$

else

$\hat{x}_i^j = \check{x}_i^j$

end if

end for

$h^j = r^j \odot m^j + 0.5(1 - r^j)$

end for

Optimize D with respect to objective

$\nabla_{\theta_d} - \sum_{j=1}^{B_D} \mathcal{L}_D(m^j, \hat{P}(\hat{x}^j, h^j))$

(2) Accuracy network optimization

Sample from the dataset $\{(x^j, m^j)\}_{j=1}^{B_A}$

Sample i.i.d., $\{z^j\}_{j=1}^{B_A}$, of Z

for $j = 1, \ldots, B_A$ **do**

$\check{x}^j \leftarrow G(\tilde{x}^j, m^j, z^j)$

for i in d **do**

if $M_i = 1$ **then**

$\hat{x}_i^j = x_i^j$

else

$\hat{x}_i^j = \check{x}_i^j$

end if

end for

end for

Optimize A with respect to objective

$\nabla_{\theta_a} - \sum_{j=1}^{B_A} \mathcal{L}_A(\hat{x}^j)$

(3) Generator optimization

Sample from the dataset $\{(\tilde{x}^j, m^j)\}_{j=1}^{B_G}$

Sample i.i.d., $\{z^j\}_{j=1}^{B_G}$, of Z

Sample i.i.d., $\{b^j\}_{j=1}^{B_G}$, of R

Optimize G with respect to objective

$\nabla_{\theta_g} \sum_{j=1}^{B_G} \alpha\mathcal{L}_M(x^j, \hat{x}^j) + \beta\mathcal{L}_G(m^j, \hat{P}(\hat{x}^j, h^j)) + \gamma\mathcal{L}_P(m^j, \hat{x}^j)$

end for

Chapter 4

Semi-supervised Learning

4.1 Semi-Supervised GACN

GACN needs all the labels to run correctly, and as shown in Chapter 5, if we do not have enough labeled samples, the GACN performance drops drastically. Therefore, an extension of GACN is presented here to address this issue. We call this algorithm semi-supervised GACN (SS-GACN). In order to achieve this goal, (3.14) and (3.18) need to be updated as follows.

$$\min_{\theta_a} -\sum_{j=1}^{B_A} \kappa^j \mathcal{L}_A(\hat{x}^j), \tag{4.1}$$

$$\min_{\theta_g} \sum_{j=1}^{B_G} \alpha \mathcal{L}_M(x^j, \hat{x}^j) + \beta \mathcal{L}_G(m^j, \hat{P}(\hat{x}^j, h^j)) + \kappa^j \gamma \mathcal{L}_P(m^j, \hat{x}^j), \tag{4.2}$$

where κ^j is a binary variable that is equal to 1 if the label for the j-th sample in the mini-batch is present and 0 otherwise. SS-GACN uses the additional information in the labels to update the classification network A for labeled samples. If a sample is unlabeled, it will still help to update the discriminator and also the generator by updating $\mathcal{L}_D$, $\mathcal{L}_M$, and $\mathcal{L}_G$. We note that SS-GACN is the most general algorithm, and both GAIN and GACN are special cases of SS-GACN, when there are no labeled samples, and all the

labels are present, respectively. In Chapter 5, we show that in the presence of partially labeled data, SS-GACN outperforms both GACN and GAIN. Algorithm 2 presents the pseudo-code of SS-GACN.

Algorithm 2 Pseudo-code of SS-GACN

for a preset number of iterations **do**

(1) Discriminator optimization

Sample from the dataset $\{(x^j, m^j)\}_{j=1}^{B_D}$

Sample i.i.d., $\{z^j\}_{j=1}^{B_D}$, of Z

Sample i.i.d., $\{r^j\}_{j=1}^{B_D}$, of R

for $j = 1, \ldots, B_D$ **do**

$\check{x}^j \leftarrow G(x^j, m^j, z^j)$

for i in d **do**

if $M_i = 1$ **then**

$\hat{x}_i^j = x_i^j$

else

$\hat{x}_i^j = \check{x}_i^j$

end if

end for

$h^j = r^j \odot m^j + 0.5(1 - r^j)$

end for

Optimize D with respect to objective

$\nabla_{\theta_d} - \sum_{j=1}^{B_D} \mathcal{L}_D(m^j, \hat{P}(\hat{x}^j, h^j))$

(2) Accuracy network optimization

Sample from the dataset $\{(x^j, m^j)\}_{j=1}^{B_A}$

Sample i.i.d., $\{z^j\}_{j=1}^{B_A}$, of Z

for $j = 1, \ldots, B_A$ **do**

$\check{x}^j \leftarrow G(\tilde{x}^j, m^j, z^j)$

for i in d **do**

if $M_i = 1$ **then**

$\hat{x}_i^j = x_i^j$

else

$\hat{x}_i^j = \check{x}_i^j$

end if

end for

end for

Optimize A with respect to objective

$\nabla_{\theta_a} - \sum_{j=1}^{B_A} \kappa^j \mathcal{L}_A(\hat{x}^j)$

(3) Generator optimization

Sample from the dataset $\{(\tilde{x}^j, m^j)\}_{j=1}^{B_G}$

Sample i.i.d., $\{z^j\}_{j=1}^{B_G}$, of Z

Sample i.i.d., $\{b^j\}_{j=1}^{B_G}$, of R

Optimize G with respect to objective

$\nabla_{\theta_g} \sum_{j=1}^{B_G} \alpha \mathcal{L}_M(x^j, \hat{x}^j) + \beta \mathcal{L}_G(m^j, \hat{P}(\hat{x}^j, h^j)) + \gamma \kappa^j \mathcal{L}_P(m^j, \hat{x}^j)$

end for

Chapter 5

Experiments

This chapter compares the performance of the proposed algorithms, GACN and SS-GACN, with that of GAIN, MICE, and mean imputation.[1] We divide our experimental results into the following sections. We begin with the MAR model for missingness. First, for the case of all labels being present, we show the effectiveness of GACN with an observation on its impact on the RMSE of imputed data. After that, we discuss the reason behind the effectiveness of GACN. Then, we show the necessity of SS-GACN in the case of partially labeled data. Then, we consider an NMAR case from the flow dataset to study the performance of GACN under this scenario. Lastly, we study the effect of GACN on other datasets.

5.1 Experimental Setup

In our experiments G and D are three-layer perceptron networks optimized with ADAM [50]. A is another three-layer perceptron, where each layer contains 30 nodes, and it is also optimized with ADAM. We choose these architectures based on extensive experimentation and hyper-parameter tuning. As mentioned in Chapter 2, the real-world

[1] We do not compare with the method in [7] since it is already shown in [36] that GAIN outperforms it.

example we consider is a combination of the ISCX VPN-nonVPN [45, 46] and ISCX Tor-nonTor [47, 48] datasets. This combination from now on is called the *flow dataset*. The flow dataset consists of 43590 encrypted TCP flows. We build a complete dataset, i.e., with no missing data, based on the flow dataset and then add artificial random missingness on our data. To build the complete dataset, we delete all the samples with at least one unobserved feature. Then, we upsample the smaller class to keep the classification fair between elements. Now with this dataset, we use 70% of the data for training, and 10% and 20% for the validation set and test set, respectively. For adding artificial missingness to the dataset, we do the following. We assume that each feature X_i is missing with probability P_{miss}. If X_i is unobserved, then $M_i = 0$. With this approach, we can design MAR cases with any desired P_{miss}.

5.2 Improved Classification Accuracy

GACN takes advantage of three sources, GAN-based loss $\mathcal{L}_G$, reconstruction loss $\mathcal{L}_M$, and classification loss $\mathcal{L}_P$. In this section, we show the effectiveness of each of these parts by experimenting with our dataset. It is notable that throughout this section, all numerical results include 90% confidence intervals, using 50 random realizations. The reported results are for test accuracy.

Figure 5.1 studies the effectiveness of having $\mathcal{L}_P$ as a part of the generator's update with P_{miss} equal to 20% and 40%. These missingness rates are common choices in this line of studies (e.g. in [8, 36]).

Figure 5.1 shows that by picking a reasonable γ, we can reach a higher accuracy faster. Figure 5.2 presents the result for different α and β values. We can conclude the importance of having reconstruction loss $\mathcal{L}_M$ since having no α drops the performance drastically.

Table 5.1 gives a comparison between GACN and GAIN for different missingness

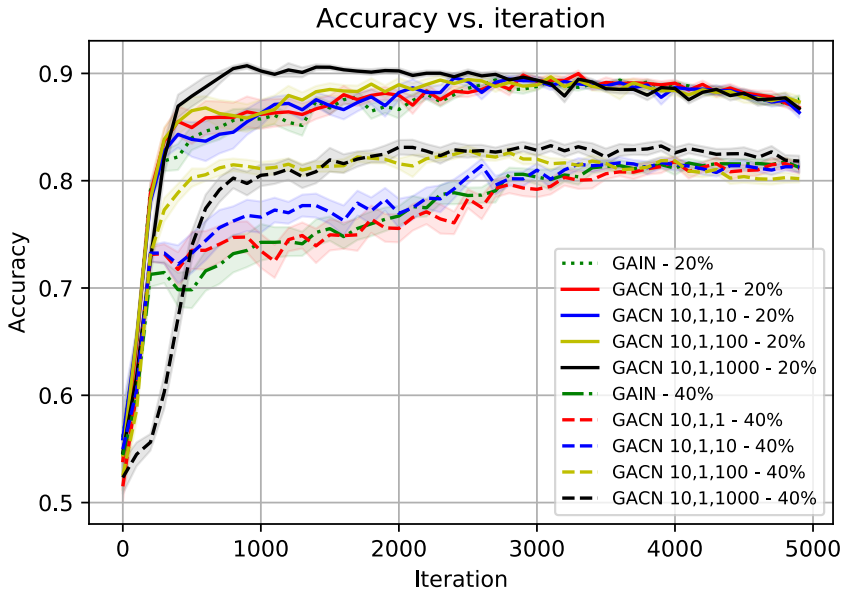

Figure 5.1: Classification accuracy vs. iteration for different γ values and different missingness rates. The three numbers in the legend are the values of α, β, γ.

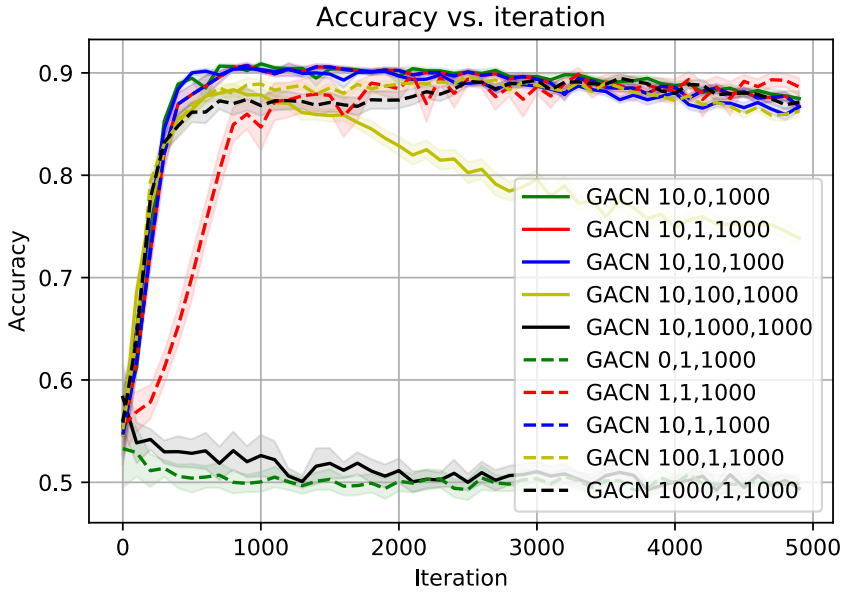

Figure 5.2: GACN classification accuracy vs. iteration for different α and β values. The three numbers in the legend are the values of α, β, γ. $P_{\text{miss}} = 20\%$.

rates. We can see that GACN always gives better classification accuracy with a tight confidence interval. In reality, we expect a dataset to have less than 50% missingness.

Missingness Rate	Accuracy GACN	Accuracy GAIN
10%	0.9376 ± 0.0036	0.9237 ± 0.0052
20%	0.9022 ± 0.0024	0.8575 ± 0.0049
30%	0.8637 ± 0.0023	0.8326 ± 0.0046
40%	0.8017 ± 0.0028	0.7258 ± 0.0100
50%	0.7307 ± 0.0079	0.6886 ± 0.0072
60%	0.6389 ± 0.0100	0.6271 ± 0.0050
70%	0.6092 ± 0.0042	0.5889 ± 0.0036
80%	0.5663 ± 0.0033	0.5462 ± 0.0043

Table 5.1: Test accuracy for GACN and GAIN at different missingness rate.

As a result, if we look at the first four rows of Table 5.1, we can see that GACN can keep a high classification accuracy until 40% missingness. This result shows the effectiveness of GACN. Table 5.2 compares the accuracy results from different imputation methods, GACN, GAIN, MICE, and mean imputation. These methods are presented in Chapter 1. All accuracies are for the test accuracy, and the number of iterations for GACN and GAIN is 1000. As we can see, GACN outperforms all the methods in terms of the test accuracy.

5.3 Imputation RMSE

Figure 5.3 shows the average RMSE for imputed features. From Figure 5.1, we concluded that GACN accuracy picks up faster than GAIN. However, in Figure 5.3, we can see that GAIN and GACN are very similar in average RMSE. To understand why GACN performs better than GAIN in terms of accuracy, we plot the average RMSE for individual features.

Algorithm	Accuracy
GACN	0.9022 ± 0.0024
GAIN	0.8575 ± 0.0049
MICE	0.8349 ± 0.0043
Mean imputation	0.7540 ± 0.0016

Table 5.2: Test accuracy for different algorithms at 20% missingness rate.

We have sorted all features based on importance. Feature importance is defined by the correlation between each feature and true labels in the complete dataset. The bigger the absolute value of the correlation coefficient, the higher the importance of the feature is. After sorting the feature based on importance, we plot the cumulative RMSE. From Figures 5.4 and 5.5, we can see that GACN tends to impute more important features more accurately. This result shows that GACN is putting an additional effort by using the term $\mathcal{L}_P$ to make sure the accuracy is high while imputing the missing data.

From Figures 5.6 and 5.7, we further show feature-based cumulative RMSE for GACN in different iterations. From these results, we can see that even though the sum RMSE at the end for GACN after iteration 90 is similar, the distribution of the per-feature RMSE values is quite different, which is why GACN gets better in terms of accuracy as the number of iteration increases.

5.4 Partially Labelled Data Imputation

In this section, we experiment with partially labeled data and show that SS-GACN can use a small amount of labeled data and still give us good performance both in terms of accuracy and imputation. Figure 5.8 compares the accuracy result for GACN, SS-GACN, and GAIN when 20% of features are missing, and 10% of the training dataset is labeled.

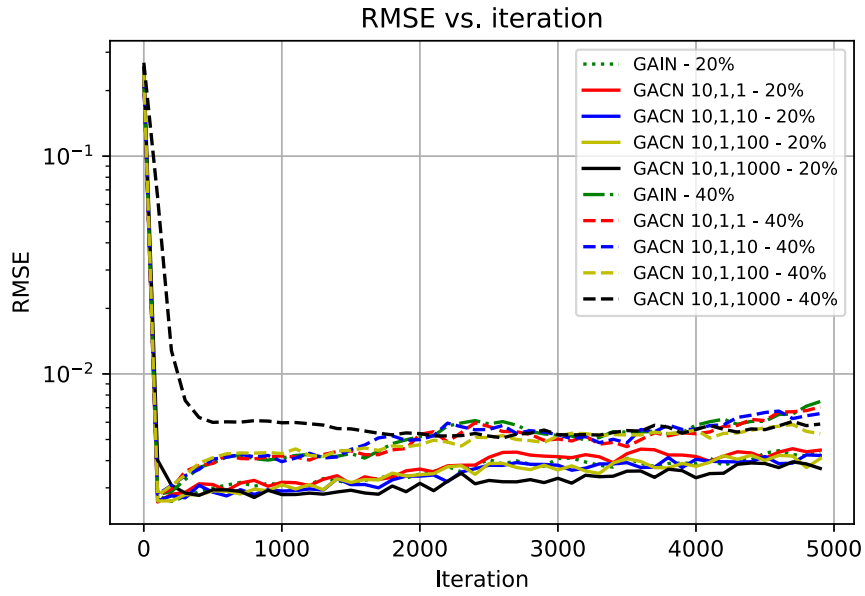

Figure 5.3: GACN RMSE vs. iteration for different γ values and different missingness rates. The three numbers in the legend are the values of α, β, γ.

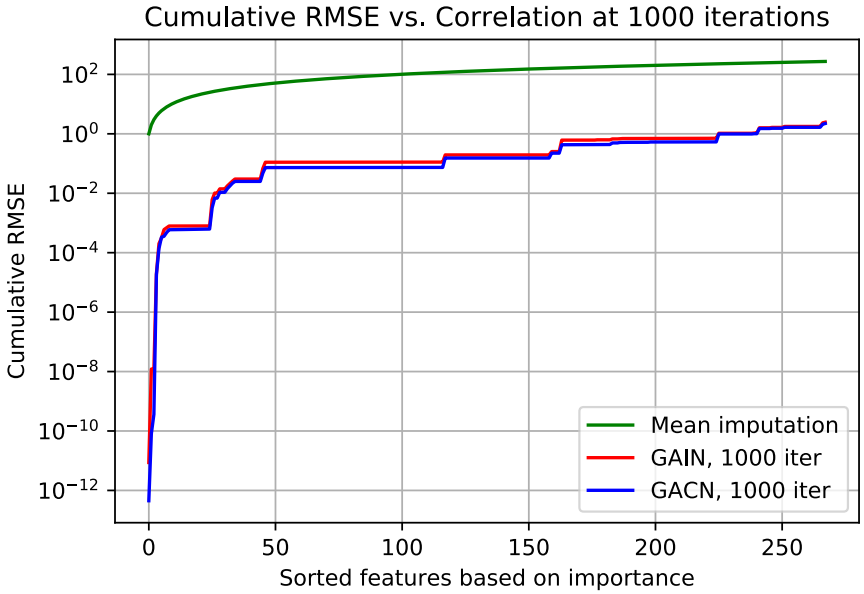

Figure 5.4: RMSE for sorted features for different algorithms.

We can see that the GACN performance drops drastically, because GACN can only use labeled data. On the other hand, GAIN preserves its performance because it does not use any labels. However, SS-GACN outperforms both of them because it can improve its

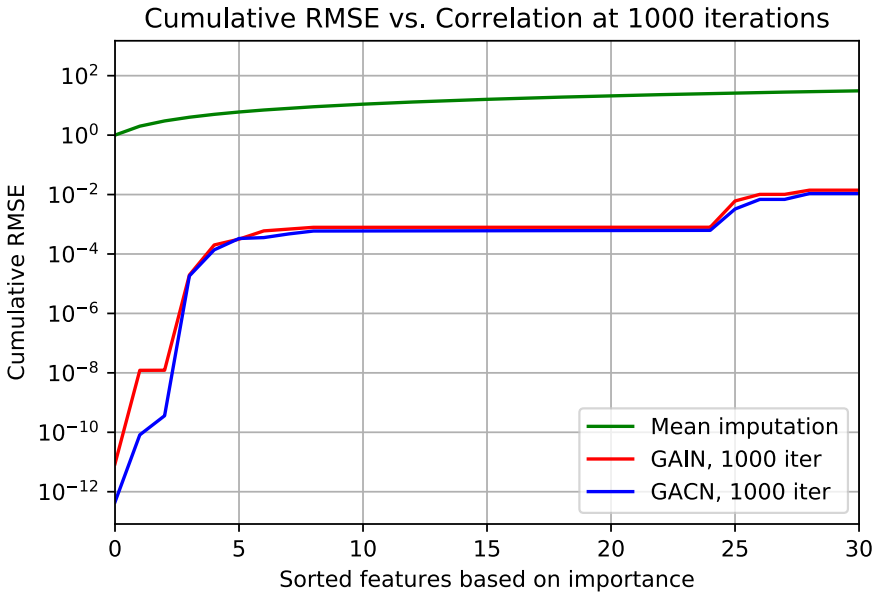

Figure 5.5: RMSE for sorted features for different algorithms, with focus on the most important features.

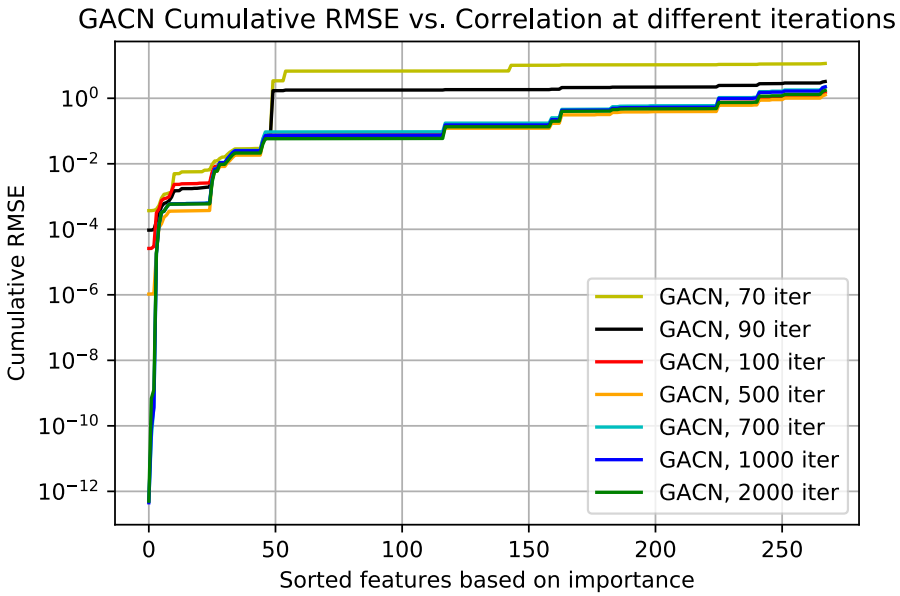

Figure 5.6: RMSE for sorted features for GACN in different iterations.

learning using the available 10% labels and use the whole dataset for imputation.

In Figure 5.9, we can see that when 40% of data are missing and 10% of them are labeled, using GACN ends in random label assignment due to the lack of samples. In

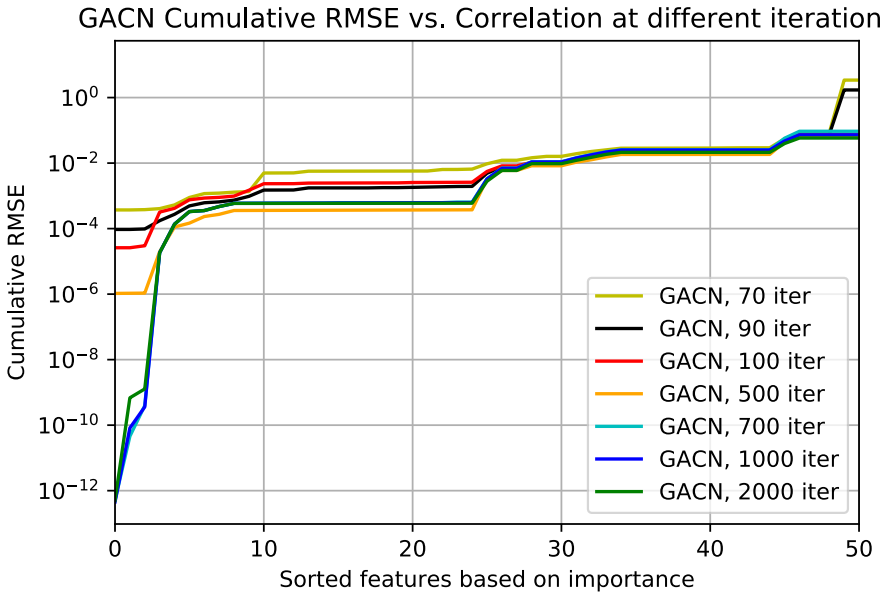

Figure 5.7: RMSE for sorted features for GACN in different iterations, with focus on the most important features.

contrast, a well-designed SS-GACN can give an accuracy rate near 80%.

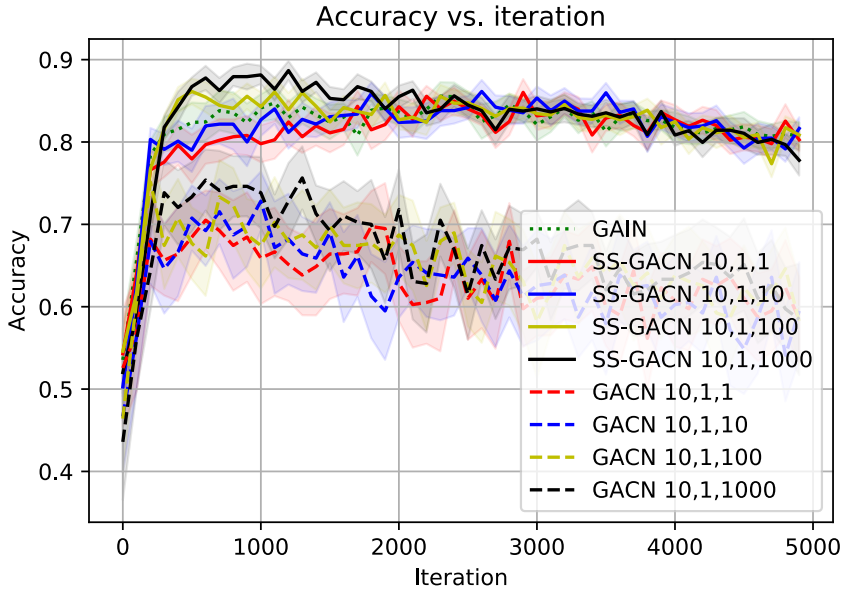

Figure 5.8: SS-GACN and GACN Classification accuracy vs. iteration for different γ values. The three numbers in the legend are the values of α, β, γ. $P_{\text{miss}} = 20\%$.

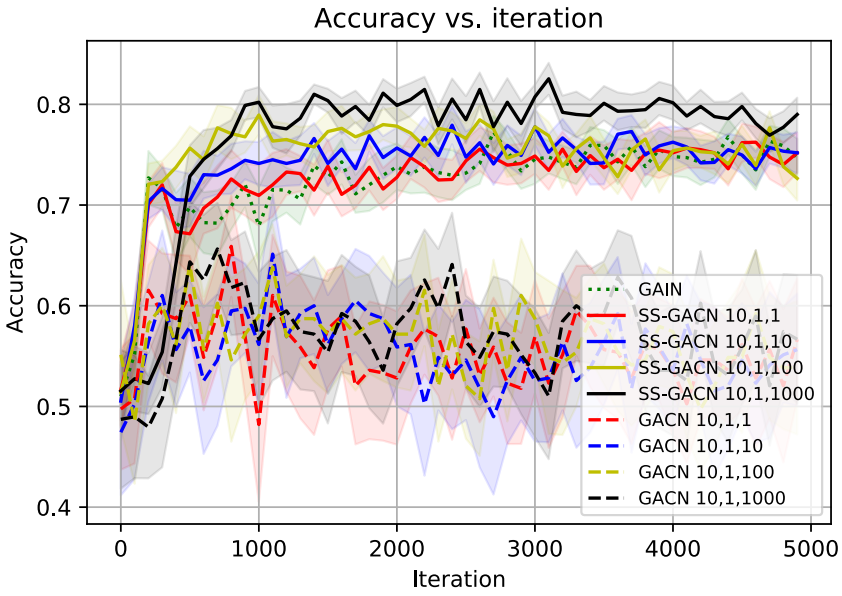

Figure 5.9: SS-GACN and GACN Classification accuracy vs. iteration for different γ values. The three numbers in the legend are the values of α, β, γ. $P_{\text{miss}} = 40\%$.

5.5 Not Missing at Random (NMAR)

Until this section, all the results were for the MAR case, where features were missing uniformly at random. As mentioned in Section 1.3.2, this assumption is popular in the literature, but it is not the case in practice. Hence, in this section, we aim to show the proposed algorithm's effectiveness for more general cases. To this end, first, in Section 5.5.1, we consider the original flow dataset and impute missing data in this real-world dataset. Later we study the effectiveness of our method with an artificial NMAR case.

5.5.1 NMAR in Flow Dataset

The original flow dataset has a 3.1% missingness rate. Furthermore, for five features, more than half of the data is missing. For this reason, we have deleted those five features from the dataset. We can see this deletion as a pair-wise deletion mentioned in Section 1.4.1. The remainder of the dataset has 2.1% missingness in it. Figure 5.10 shows the results for

imputing missing data in this dataset. We observe that GACN maintains its performance advantage of GAIN.

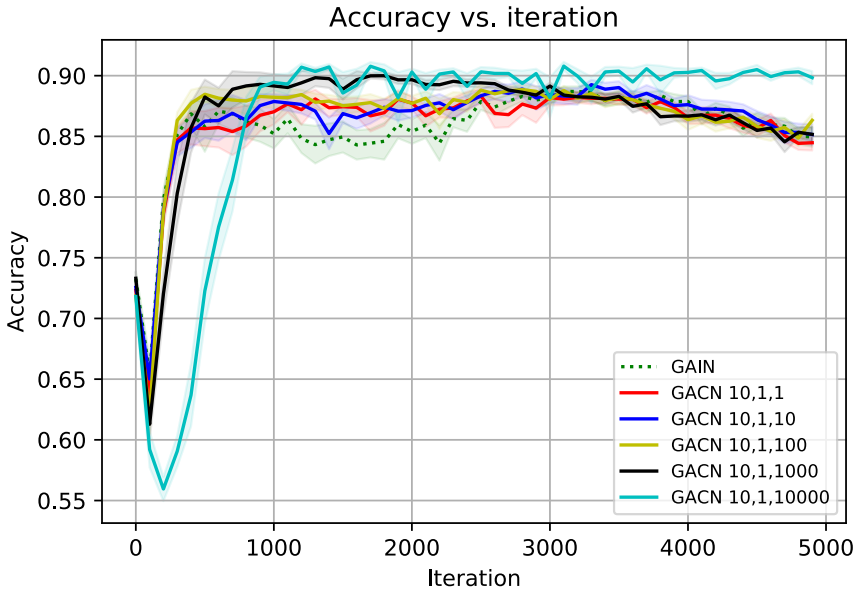

Figure 5.10: Real NMAR classification accuracy vs. iteration for different γ values. The three numbers in the legend are the values of α, β, γ.

5.5.2 Artificial NMAR

For building an artificial NMAR case, we consider the same complete flow dataset from Section 5.1. This time for adding missingness to the dataset, we look at the labels. The total P_{miss} is still 20%. However, in this case, 30% delay-tolerant data entries are missing, while only 10% of the delay-sensitive examples are missing. This is an NMAR case because missingness depends on the value of labels. Figure 5.11 shows the result of this study. We still observe substantial performance improvement from GACN.

5.6 Other Datasets

In this part, we consider the performance of GACN on the Spam and Credit datasets from the UCI Machine Learning Repository [51]. In Figure 5.12 we show the results for

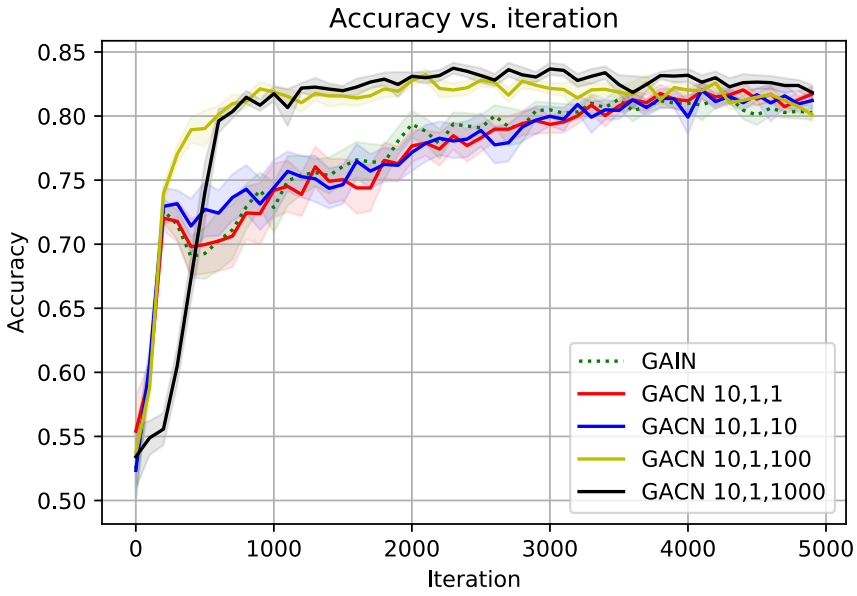

Figure 5.11: Artificial NMAR classification accuracy vs. iteration for different γ values. The three numbers in the legend are the values of α, β, γ.

the Spam dataset. Figure 5.13 shows the experimental results for the so-called modified credit dataset, where only continuous features are kept. What is interesting in these results is that here we do not see the considerable improvement GACN has on the flow dataset.

To investigate why GACN is not giving substantial improvement, we further study the difference in the characteristics of these datasets. To this end, we study the feature importance once again. We plot the correlation between individual features and labels and sort the values in descending order. Figures 5.14, 5.15, and 5.16 show the correlation between labels and individual features for the flow dataset, the Spam dataset, and the Credit dataset, respectively. From Figure 5.14, we can see that some features have a substantially higher correlation for the flow dataset compared with the others. For 12.8% of the features, we have a correlation value greater than 0.1. For the most important 34 features, the curve slope is substantially steeper than the rest of the graph. GACN is able to understand this substantial difference and learn which features are more important. This knowledge makes sure the algorithm puts enough effort into learning these critical

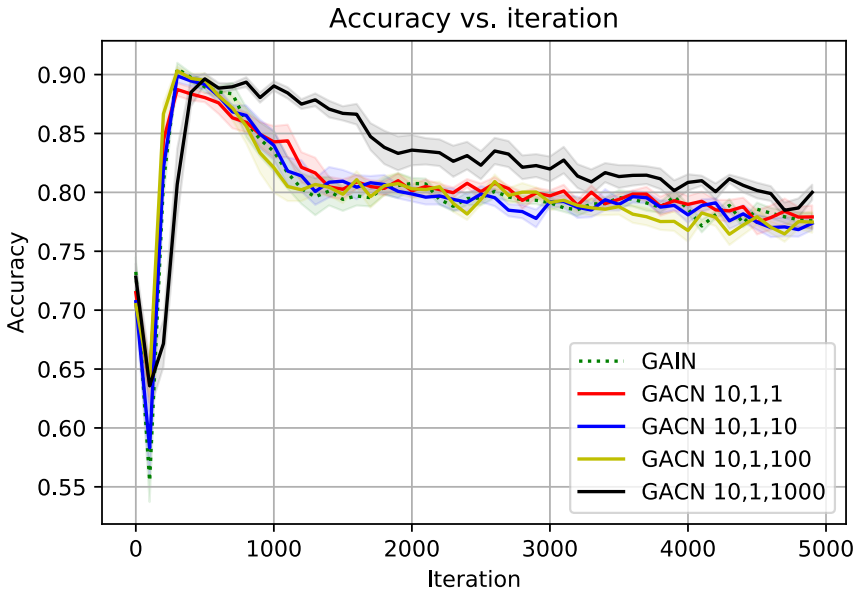

Figure 5.12: Classification accuracy vs. iteration for different γ values. Spam dataset. The three numbers in the legend are the values of α, β, γ.

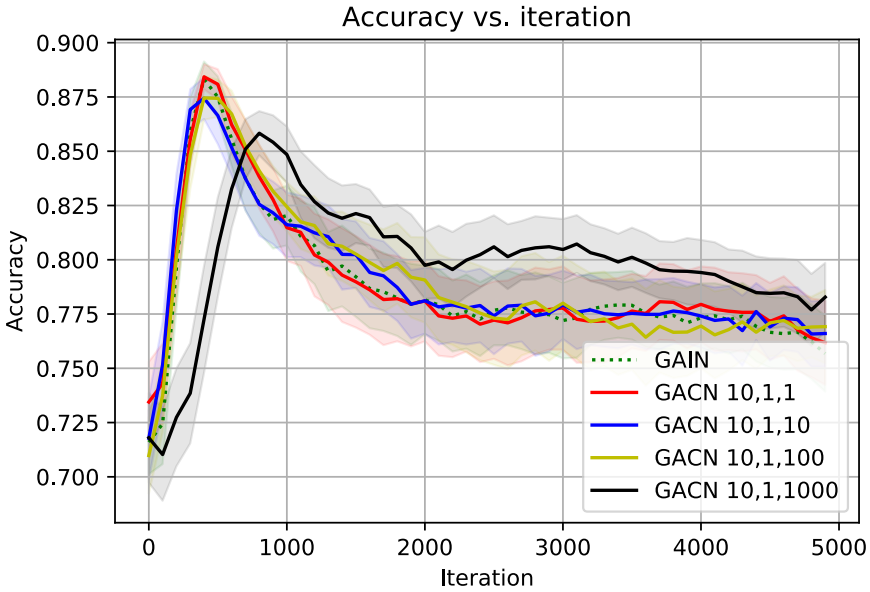

Figure 5.13: Classification accuracy vs. iteration for different γ values. Modified credit dataset. The three numbers in the legend are the values of α, β, γ.

features in terms of accuracy. From Figures 5.15 and 5.16, we can see a more gradual change in the value of the correlation. Unlike the flow dataset, here, there is no subset

of the dataset that has a substantially higher impact on accuracy. This is why GAIN is as good as GACN for these datasets.

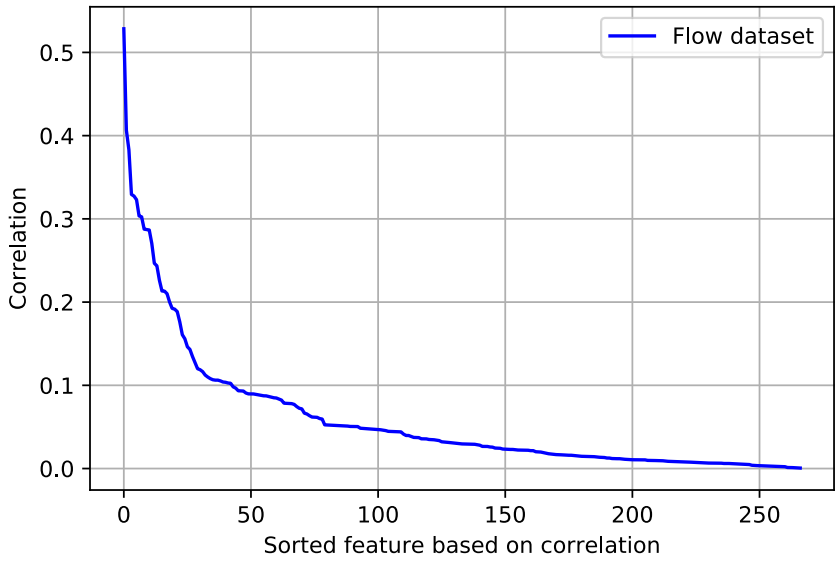

Figure 5.14: Correlation between labels and individual features. For Flow dataset.

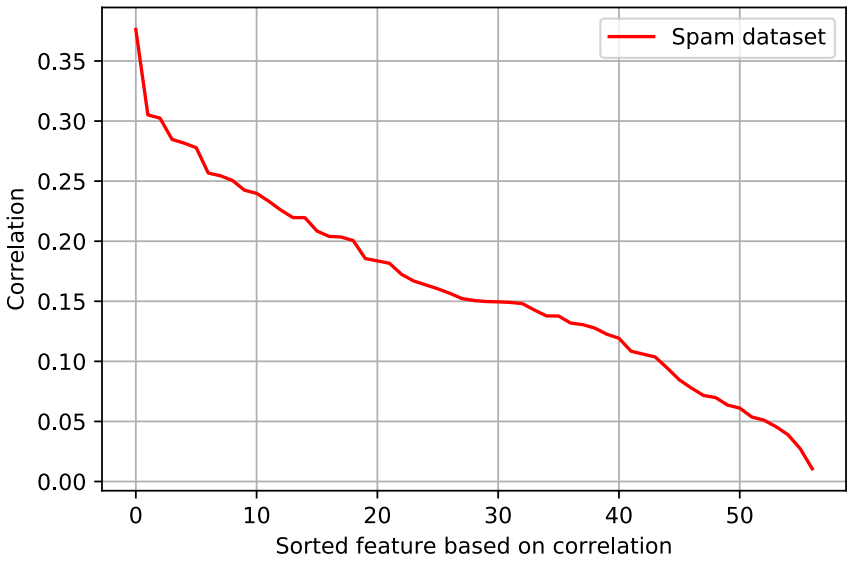

Figure 5.15: Correlation between labels and individual features. For Spam dataset.

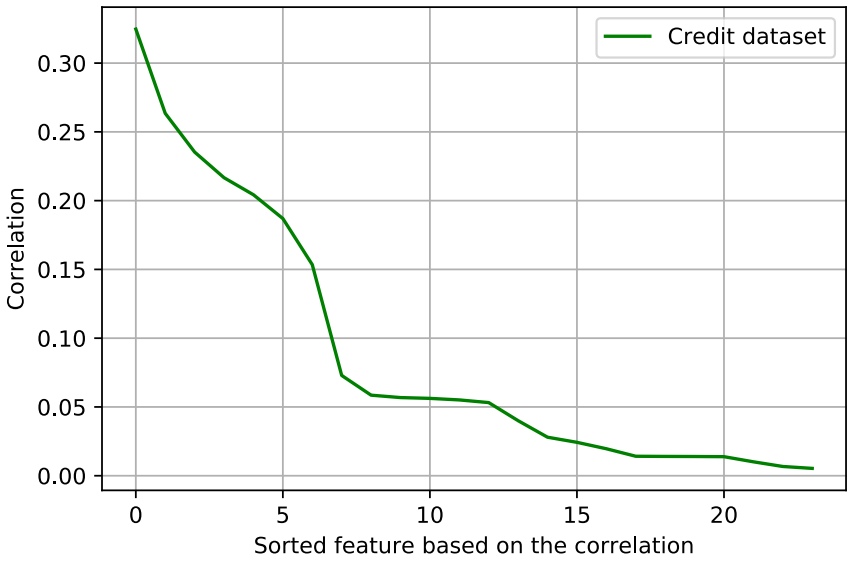

Figure 5.16: Correlation between labels and individual features. For Credit dataset.

Chapter 6

Conclusion and Future Directions

The imperfection of real-world datasets causes many challenges in machine learning and statistical analysis. In this thesis, we bring attention to the problem of classification under the existence of missing data. We propose a new approach for data imputation based on the classification accuracy gain. In this final chapter of the thesis, we will indicate the concluding points and present future directions.

6.1 Conclusion

This work starts with a survey on the literature of missing data. We categorize these work by their applications, research methods, and models of missingness. We further give details of important approaches for handling missing data. We conclude that missing data imputation is still an open problem. Specifically, past imputation methods are insufficient when the main objective is classification accuracy.

After showing a need for a new imputation method that performs better in terms of classification accuracy, we propose a novel deep generative method that takes classification accuracy into account. The proposal estimates missing data values while taking the classification accuracy into account. Our method, called generative adversarial classification method (GACN), consists of three neural networks working together to learn the

data distribution, impute missing values, and compute classification accuracy.

Our experimental results show the performance of GACN under different scenarios. We conclude that our proposal achieves better classification performance compared with state-of-the-art methods. We investigate the reason behind GACN's better performance. We conclude that GACN has the advantage of learning which feature is more important for the final classification accuracy. With this knowledge, GACN is able to impute missing values of more important features more accurately.

We further expand our proposal for partially labeled datasets. We call this variation of GACN semi-supervised GACN (SS-GACN). SS-GACN is able to use both unlabeled and labeled data entries. This makes SS-GACN useful in applications where it is hard to collect all the labels. Empirical results show that for datasets in which only a small percentage of data is labeled, SS-GACN maintains classification accuracy at an acceptable level.

6.2 Future Directions

In this section, we present some of the potentially interesting extensions of this work. One of the interesting paths to follow is the NMAR cases. There is a need to understand what is the limit of being able to handle NMAR cases. The idea is to understand how different missingness structure can affect the performance of the proposed method. Ultimately, it may be possible to find a bound or a limit on how well a method can perform under different missingness distributions.

Another interesting problem that is not limited to GACN, GAIN, or even GAN, and is a general question on any nonconvex-nonconave game setting, is the convergence when using a gradient approach. Finding a suitable replacement for this approach will solve an important problem.

This thesis considers tabular datasets, which is one of the most important data types.

Another avenue for further research is an extension of this model for other types of datasets. An example is a time series. Many datasets in healthcare and finances are in this format. This work can be extended for application on time series. This means presenting an equivalent model for imputing missing values in time series considering the time-series labels.

Bibliography

[1] B. Marlin, "Missing data problems in machine learning," PhD dissertation, University of Toronto, Toronto, ON, Canada, 2008.

[2] D. J. Poirier, and P. A. Ruud, "Diagnostic testing in missing data models", *International Economic Review*, vol. 24, no.3, pp. 537-546, 1983.

[3] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for DNA microarrays", *Bioinformatics*, vol. 17, pp. 520-525, 2001.

[4] P. L. Roth, "Missing data: A conceptual review for applied psychologists", *Personnel Psychology*, vol. 47, no. 3, pp. 537-561, 1994.

[5] C. K. I. Williams, C. Nash, and A. Nazabal, "Autoencoders and probabilistic inference with missing data: An exact solution for the factor analysis case", arXiv:1801.03851, 2018.

[6] B. M. Marlin, S. T. Roweis, and R. S. Zemel, "Unsupervised learning with non-ignorable missing data", in Proceedings of *The Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT)*, 2005, pp. 222-229.

[7] P. M. Comar, L. Liu, S. Saha, P. N. Tan, and A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection", in Proceedings of *INFOCOM*, 2013, pp. 2022-2030.

[8] A. Nazabal, P. Olmos, Z. Ghahramani, I. Valera, "handling incomplete heterogeneous data using VAEs", arXiv:1807.03653, 2018.

[9] R. D. Camino, C. A. Hammerschmidt, and R. State, "Improving missing data imputation with deep generative models", arXiv: 1902.10666, 2019.

[10] F. Biessmann, T. Rukat, P. Schmidt, P. Naidu, S. Schelter, A. Taptunov, D. Lange, and D. Salinas, "DataWig: Missing Value Imputation for Tables", *Journal of Machine Learning Research*, vol. 20, no. 175, pp. 1-6, 2019.

[11] D. B. Rubin, "Multiple imputation for nonresponse in surveys", *John Wiley and Sons*, vol. 81, 2004.

[12] H. Kim, G. H. Golub, and H. Park, "Missing value estimation for DNA microarray gene expression data: Local least squares imputation," *Bioinformatics*, vol. 21, pp. 187-198, 2005.

[13] P. Hayati Rezvan, K. J. Lee, J. A. Simpson "The rise of multiple imputation: a review of the reporting and implementation of the method in medical research", *BMC Medical Research Methodology*, 2015.

[14] S. Dray and J. Josse, "Principal component analysis with missing values: a comparative survey of methods", *Plant Ecology*, vol. 216, pp. 657-667, 2015.

[15] S. G. Heeringa, and R. L. Woodburn "The 1989 surveys of consumer finances sample design documentation", *Survey Research Center, University of Michigan, Ann Arbor*, 1991.

[16] P. L. Roth, F. S. Switzer III, D. Switzer, "Missing data in multi-item scales: A Monte Carlo analysis of missing data techniques", *Organizational Research Methods*, vol. 2, no. 3, pp. 211-232, 1999.

[17] Y. Li, Z. Li, and L. Li, "Missing traffic data: comparison of imputation methods", *IET Intelligent Transport Systems*, vol. 8, pp. 51-57, 2014.

[18] N. Zhao, Z. Li, and Y. Li, "Improving the traffic data imputation accuracy using temporal and spatial information," in Proceedings of *International Conference on Intelligent Computation Technology and Automation*, 2014, pp. 312-317.

[19] H. Chen, S. Grant-Muller, L. Mussone, and F. Montgomery, "A study of hybrid neural network approaches and the effects of missingdata on traffic forecasting", *Neural Computing and Applications*, vol. 10, pp. 277-286, 2001.

[20] A. Mottini, A. Lheritier, and R. Acuna-Agost, "Airline passenger name record generation using generative adversarial networks", arXiv:1807.06657, 2018.

[21] A. Kokaram, "On missing data treatment for degraded video and film archives: A survey and a new Bayesian approach," *IEEE Transaction on Image Processing*, vol. 13, no. 3, pp. 397-415, March 2004.

[22] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks", in Proceedings of *Advances in Neural Information Processing Systems (NeurIPS)*, 2012, pp. 350-358.

[23] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous structure and texture image inpainting", *IEEE Transaction on Image Processing*, vol. 12, no. 8, pp. 882-889, 2003.

[24] S. Ahmed and V. Tresp, "Some solutions to the missing feature problem in vision", in Proceedings of *Advances in Neural Information Processing Systems (NeurIPS)*, 1993, pp. 393-400.

[25] V. Tresp, R. Neuneier, and S. Ahmed, "Efficient methods for dealing with missing data in supervised learning", in Proceedings of *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 689-696, 1995.

[26] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral Regularization Algorithms for Learning Large Incomplete Matrices," *Journal of Machine Learning Research*, vol. 11, pp. 2287-2322, 2010.

[27] A. Ilin and T. Raiko, "Practical approaches to principal component analysis in the presence of missing data", *Journal of Machine Learning Research*, pp. 1957-2000, 2010.

[28] T. Tabouy, P. Barbillon, and J. Chiquet, "Variational inference for stochastic block models from sampled data", *Journal of the American Statistical Association*, pp. 1-20, 2019.

[29] M. Jamshidian, P. M. Bentler, "ML Estimation of Mean and Covariance Structures with Missing Data Using Complete Data Routines", *Journal of Educational and Behavioral Statistics*, pp. 21-41, 1999.

[30] S. Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r", *Journal of statistical software*, 2011.

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

[32] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an EM approach", in Proceedings of *Advances in Neural Information Processing Systems (NeurIPS)*, pages 120-127, 1994.

[33] D. J. Stekhoven, P. Buhlmann, "MissForest-non-parametric missing value imputation for mixed-type data", *Bioinformatics* 2012, pp. 112-118.

[34] T. Fang and A. Schwing, : "Co-generation with gans using ais based hmc," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[35] L. Gondara and K. Wang, "MIDA: multiple imputation using deep denoising autoencoders", arXiv:1705.02737, 2017.

[36] J. Yoon, J. Jordon, and M. van der Schaar, "GAIN: Missing data imputation using generative adversarial nets", in Proceedings of *International Conference on Machine Learning (ICML)*, pp. 5689–5698, 2018.

[37] Y. Luo, X. Cai, Y. Zhang, J. Xu, and Yuan Xiaojie, "Multivariate time series imputation with generative adversarial networks," in Proceedings of *Advances in Neural Information Processing System (NeurIPS*, 2018, pp. 1596–1607.

[38] V. Fortuin, G. Ratsch, and S. Mandt, "GP-VAE: Deep probabilistic time series imputation", *arXiv:1907.04155*, 2019.

[39] Y. Liu, R. Yu, S. Zheng, E. Zhan, and Y. Yue, "Naomi: Non-autoregressive multiresolution sequence imputation," in Proceedings of *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[40] P. Mattei and J. Frellsen, "MIWAE: Deep generative modelling and imputation of incomplete data sets", in Proceedings of *International Conference on Machine Learning (ICML)*, 2019.

[41] P. E. Cheng, "Nonparametric estimation of mean functionals with data missing at random", *Journal of American Statistics Association*, pp. 81-87, 1994.

[42] U. Hwang, D. Jung, and S. Yoon, "HexaGAN: Generative Adversarial Nets for Real World Classification", *arXiv:1902.09913*, 2019.

[43] R. J. A. Little and D. B. Rubin, "Statistical analysis with missing data", *John Wiley and Sons, Inc*, 1987.

[44] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. WardeFarley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets", in Proceedings of *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[45] 2016. ISCX TOR-nonTOR. Available online: https://www.unb.ca/cic/datasets/tor.html.

[46] G. Draper-Gil, A. Lashkari, M. Mamun, and A. Ghorbani, "Characterization of Encrypted and VPN Traffic using Time-related Features", in Proceedings of *2nd Int. Conf. on Inf. Sys. Security and Privacy*, pp. 407–414, 2016.

[47] 2016. ISCX VPN-nonVPN. Available online: https://www.unb.ca/cic/datasets/vpn.html.

[48] A. Lashkari, G. Draper-Gil, M. Mamun, and A. Ghorbani, "Characterization of Tor Traffic using Time based Features", in Proceedings of *3rd Int. Conf. on Inf. Sys. Security and Privacy*, 2017.

[49] S. Chowdhury, B. Liang, and A. Tizghadam, "Explaining class-of-service oriented network traffic classification with superfeatures," in Proceedings of *ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks (Big-DAMA)*, 2019.

[50] D. Kingma and J. Ba. "Adam: A method for stochastic optimization," in Proceedings of *International Conference on Learning Representations (ICLR)*, 2015.

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