Adaptive Data Augmentation for Supervised Learning over Missing Data

Tongyu Liu
Renmin University of China
ltyzzz@ruc.edu.cn

Yinqing Luo
Renmin University of China
lyqroy@ruc.edu.cn

Ju Fan
Renmin University of China
fanj@ruc.edu.cn

Nan Tang
Qatar Computing Research Institute
ntang@hbku.edu.qa

Guoliang Li
Tsinghua University
liguoliang@tsinghua.edu.cn

Xiaoyong Du
Renmin University of China
duyong@ruc.edu.cn

ABSTRACT

Real-world data is dirty, which causes serious problems in (supervised) machine learning (ML). The widely used practice in such scenario is to first repair the labeled source (a.k.a. train) data using rule-, statistical- or ML-based methods and then use the "repaired" source to train an ML model. During production, unlabeled target (a.k.a. test) data will also be repaired, and is then fed in the trained ML model for prediction. However, this process often causes a performance degradation when the source and target datasets are dirty with different *noise patterns*, which is common in practice.

In this paper, we propose an *adaptive data augmentation* approach, for handling missing data in supervised ML. The approach extracts noise patterns from target data, and adapts the source data with the extracted target noise patterns while still preserving supervision signals in the source. Then, it *patches* the ML model by retraining it on the adapted data, in order to better serve the target. To effectively support adaptive data augmentation, we propose a novel generative adversarial network (GAN) based framework, called DAGAN, which works in an unsupervised fashion. DAGAN consists of two connected GAN networks. The first GAN learns the noise pattern from the target, for *target mask generation*. The second GAN uses the learned target mask to augment the source data, for *source data adaptation*. The augmented source data is used to retrain the ML model. Extensive experiments show that our method significantly improves the ML model performance and is more robust than the state-of-the-art missing data imputation solutions for handling datasets with different missing value patterns.

1 INTRODUCTION

Machine learning (ML) techniques have been deployed in (almost) all applications nowadays. Among all the techniques, supervised ML, such as classification and regression, is the most prevalent [39]. Typically, a supervised ML application follows two basic steps. First, data scientists or ML engineers prepare a labeled *source* dataset (a.k.a. training dataset) and train a supervised learning model, e.g., a classifier, on the dataset. Then, the trained model will be deployed in one or many production environments to make prediction on the *target* datasets (a.k.a. test datasets) [28].

In practice, unfortunately, the ML models are difficult to maintain in the production environments. One central challenge are *unexpected errors* in the target data, which is fed into the ML models at prediction time [35, 42]. Among all the data errors [6], missing data [41] is a serious problem that data scientists need to account for everyday. There are a multitude of reasons why they occur: ranging from human errors during data entry, incorrect sensor readings, to software bugs in data science pipelines [16]. Moreover, different from other types of errors (e.g., wrong names/addresses, unnormalized values, and violations of integrity constraints) that sometimes can be remained as they are, for ML modeling, these missing data fields *must* be deleted or imputed first.

(a) A labeled source data ($\mathcal{D}_s$) labels

	age	chol	gluc	smoke	alcohol	cardio
s_1	30	2	3	0	0	no
s_2	35	2	1	1	0	no
s_3	50	3	3	NA	1	yes
s_4	65	2	3	1	NA	yes
s_5	70	3	1	1	0	no

(b) An unlabeled target data ($\mathcal{D}_t$) no labels

	age	chol	gluc	smoke	alcohol	cardio
t_1	25	1	NA	0	1	?
t_2	37	NA	3	0	0	?
t_3	40	3	NA	1	1	?
t_4	72	3	2	0	0	?

Figure 1: Source and target with different missing patterns.

Consider an exemplary scenario as shown in Figure 1. Suppose that a hospital trains a classifier that predicts cardiovascular disease (i.e., `cardio`) for patients based on a labeled source dataset $\mathcal{D}_\mathrm{s}$, which contains examination features, such as cholesterol (`chol`) and glucose (`gluc`), patient-reported features, such as smoking (`smoke`) and alcohol intake (`alcohol`), and demographics of patients, such as `age`. We can observe that $\mathcal{D}_\mathrm{s}$ contains missing values in attributes `smoke` and `alcohol`, possibly because some patients may not want to report their habits. However, when being deployed in a production environment for prediction, the missing pattern of the unlabeled target data $\mathcal{D}_\mathrm{t}$ might be different, as shown in Figure 1(b). There could be many reasons for such noise shift. For example, the model is deployed to predict another patient cohort or even in another hospital, where patients have missing values in examination features instead of smoking or alcohol habits. Not surprisingly, the model performance often degrades significantly when encountering the *noise shift* in the target data.

Limitations of Missing Data Imputation Methods. To tackle the problem, the existing works have proposed many methods for *imputing missing values* [34], using mean imputation, regression imputation, maximum likelihood, multiple imputation, etc. Ideally, if both $\mathcal{D}_\mathrm{s}$ and $\mathcal{D}_\mathrm{t}$ can be imputed to be their ground-truth, we can solve the problem. In practice, unfortunately, this is extremely hard, because of the high cost of data cleaning [6]. Thus, in most cases, practitioners will pick data imputation methods from the decades-long effort on this field [41]. However, regardless of how sophisticated data cleaning techniques the practitioners may choose, these techniques may have suboptimal results, i.e., the repaired data is not the ground-truth. Subsequently, data imputation separately for $\mathcal{D}_\mathtt{s}$ and $\mathcal{D}_\mathtt{t}$ might cause an even bigger *divergence* on data distributions, thus degrading the model performance. Consider our previous example shown in Figure 1: an imputation method may lead to a result that distribution of attribute gluc in target $\mathcal{D}_\mathtt{t}$ diverges from that in source $\mathcal{D}_\mathtt{s}$. This would result in a well-recognized problem in ML, called *dataset shift* [9, 23, 42], which may severely affect the performance of ML models.

Our Solution. Because finding the ground-truth for $\mathcal{D}_\mathtt{s}$ and $\mathcal{D}_\mathtt{t}$ is practically infeasible, one alternative is to extract some "signal" (or noise pattern) from the target, which can be used to adapt the source and retrain the model, in order to better serve the target.

Based on this idea, we introduce a novel approach, dubbed *adaptive data augmentation*. It first adapts source data $\mathcal{D}_\mathtt{s}$ to $\tilde{\mathcal{D}}$ by learning missing patterns from an unlabeled dataset in the target and then "patches" the ML model by retraining the model on $\tilde{\mathcal{D}}$.

The objective of the adaptation is two-fold. First, the adapted dataset $\tilde{\mathcal{D}}$ should preserve the supervision signals in $\mathcal{D}_\mathtt{s}$, e.g., the conditional feature distribution with respect to label remains constant. For example, we do not want $\tilde{\mathcal{D}}$ to contain tuples that violate the potential correlation between features and the cardio outcome. Second, the adapted dataset $\tilde{\mathcal{D}}$ should be similar to our target dataset $\mathcal{D}_\mathtt{t}$. As we consider the case that source and target have the same data distribution but different missing patterns, this essentially means that the missing pattern between $\tilde{\mathcal{D}}$ and $\mathcal{D}_\mathtt{t}$ should be as similar as possible.

The main challenges are to learn the (missing) noise pattern from the target and adapt it to the source. We propose a novel generative adversarial network (GAN) [18] based framework, namely DAGAN. DAGAN is an end-to-end learning approach that consists of two connected GAN networks. (i) *Target mask generation*: The first GAN is to learn the noise pattern from the target, e.g., younger people (i.e., age ≤ 40) tend to have more incomplete examination results in attributes chol and gluc as shown in Figure 1. (ii) *Source data adaptation*: The second GAN uses the learned target mask to "translate" the source data to an adapted data, which is indistinguishable from the real target data. Another challenge is to apply the adapted data to improve the ML model. We retrain the model using the adapted data and get a "patched" model, to better serve the target data $\mathcal{D}_\mathtt{t}$. We also study a more practical scenario with multiple dirty target datasets $\{\mathcal{D}_\mathtt{t}^1, \ldots, \mathcal{D}_\mathtt{t}^m\}$, each with a different noise pattern. We propose a method to retrain one single patched model for all targets, by using Group Distributionally Robust Optimization (GDRO) [40].

Contributions. We make the following notable contributions.

(1) We propose a novel framework to adaptively augment the source data to be alike the target data and retrain an ML model using the augmented source data, which can better serve the prediction on the unseen target data (Sections 2 and 3).

(2) We propose a novel GAN-based approach DAGAN for adaptive data augmentation (Section 4) and introduce effective model retraining methods for patching the supervised ML model (Section 5).

(3) We experiment on real-world datasets and show that our adaptive data augmentation using DAGAN can improve the model performance, and is robust for different missing data patterns, including missing not at random (MNAR), missing at random (MAR), and missing completely at random (MCAR) (Section 6).

2 PRELIMINARIES

Data Model. We consider a relational table $\mathcal{D}$ with attributes $\mathcal{A} = \{A_1, A_2, \ldots, A_N\}$ and with tuples $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(M)}\}$. In particular, we consider both categorical (nominal) and numerical (either discrete or continuous) attributes. We use x_j to denote the value of attribute A_j in tuple $\boldsymbol{x}$ (i.e., $x_j = \boldsymbol{x}[A_j]$).

Missing Data. We consider that $\mathcal{D}$ contains *missing values*: each tuple $\boldsymbol{x}$ may have some attributes that are not observed. To formalize these missing values, like the existing works [22, 50], we introduce a *mask* vector $\boldsymbol{m} \in \{0, 1\}^N$ to indicate which attributes in $\boldsymbol{x}$ are observed (i.e., not missing): $x_j \in \boldsymbol{x}$ is observed if $m_j = 1$ (m_j is the j-th entry of $\boldsymbol{m}$), otherwise x_j is missing and is denoted as NA for ease of presentation. Thus, our dataset $\mathcal{D}$ can be represented as $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{m}^{(i)})\}$. For a clearer notation, we use $\boldsymbol{x}_g$ to denote the complete (i.e., ground-truth) tuple without missing values, and $\boldsymbol{x}$ to denoted the observed incomplete tuple, and we have:

$$\boldsymbol{x} = \psi(\boldsymbol{x}_g, \boldsymbol{m}) = \boldsymbol{x}_g \odot \boldsymbol{m} + \mathtt{NA} \cdot (1 - \boldsymbol{m}), \tag{1}$$

where $\odot$ is element-wise multiplication and $(1 - \boldsymbol{m})$ is the complement of $\boldsymbol{m}$. We consider three common missing mechanisms [50] to model the conditional distribution $p(\boldsymbol{m}|\boldsymbol{x}_g)$ of mask given the ground-truth data. (1) *Missing completely at random (MCAR)*: mask $\boldsymbol{m}$ is *independent* with the data, i.e., $p(\boldsymbol{m}|\boldsymbol{x}_g) = p(\boldsymbol{m})$; (2) *Missing at random (MAR)*: mask $\boldsymbol{m}$ is only dependent on the *observed* data, i.e., $p(\boldsymbol{m}|\boldsymbol{x}_g) = p(\boldsymbol{m}|\boldsymbol{x})$; (3) *Missing not at random (MNAR)*: mask $\boldsymbol{m}$ is dependent on both *observed* and *unobserved* data.

Example 1. *Consider Figure 1. We can observe that some tuples contain missing values. For example, tuple s_3 is incomplete as value of* smoke *is missing. Thus, s_3 corresponds to a mask vector $\boldsymbol{m} = [1, 1, 1, 0, 1]$. Suppose that the ground-truth of s_3 is $\boldsymbol{x}_g = (50, 3, 3, 1, 1)$. We represent the observed tuple corresponding to s_3 as $\boldsymbol{x} = \boldsymbol{x}_g \odot \boldsymbol{m} + \mathtt{NA} \cdot (1 - \boldsymbol{m}) = (50, 3, 3, \mathtt{NA}, 1)$.*

Remarks. In practice, one may use various feature generation methods to compute a numerical representation of $\mathcal{D}$, and the dimension of $\boldsymbol{x}$ would be larger than the attribute number N. For example, we can use *one-hot* encoding for categorical attributes, e.g., representing the values of chol as vectors $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$. In this case, we will slightly modify Equation (1) to support *attribute-wise* masking, i.e., retaining entries of $\boldsymbol{x}$ corresponding to an attribute, or changing them all into NA. In this paper, for simplicity, we consider that the dimension of $\boldsymbol{x}$ is N.

Supervised ML on Missing Data. Without loss of generality, we consider a classification model $f : X \to Y$ that maps input tuple $\boldsymbol{x}^{(i)}$ in an input space X to a label from $Y = \{0, 1, \ldots, L - 1\}$. A typical classification pipeline consists of the following two steps. The first step is to train the classification model f on *labeled source* data $\mathcal{D}_\mathtt{s} = \{(\boldsymbol{x}_\mathtt{s}^{(i)}, \boldsymbol{m}_\mathtt{s}^{(i)}, y^{(i)})\}$, where $y^{(i)}$ and $\boldsymbol{m}_\mathtt{s}^{(i)}$ are the label and mask vector of tuple $\boldsymbol{x}_\mathtt{s}^{(i)}$ respectively. The second step is to deploy the model f in a production environment to serve some *unlabeled target* data $\mathcal{D}_t = \{(\boldsymbol{x}_t^{(i)}, \boldsymbol{m}_t^{(i)})\}$.

In practice, unfortunately, the performance of model f often degrades significantly on the target data [32], caused by many different real-world scenarios: (1) $\mathcal{D}_s$ and $\mathcal{D}_t$ come from different domains; (2) $\mathcal{D}_s$ and $\mathcal{D}_t$ come from the same domain but with different data distributions; or (3) $\mathcal{D}_s$ and $\mathcal{D}_t$ come from the same domain, with the same (or similar) data distributions, but have different noise (e.g., missing data) patterns; that is, there exists a *noise shift* other than a *data shift* between the source and the target. The common way to combat (1) is through transfer learning [8, 49] or domain adaptation [17], and to deal with (2) is by collecting more training examples, e.g., through active learning [28].

In this paper, we focus on case (3) by considering that source $\mathcal{D}_s$ and target $\mathcal{D}_t$ come from the same (ground-truth) data distribution, but have different *dirtiness*, i.e., the presence of missing values in $\mathcal{D}_s$ and $\mathcal{D}_t$ have different patterns. This is a very common scenario in practice, especially when $\mathcal{D}_s$ and $\mathcal{D}_t$ are from different departments or prepared by different engineering teams. Formally, we consider the masks $\boldsymbol{m}_s$ and $\boldsymbol{m}_t$ are from different *mask distributions*, while the ground-truth data is from the same data distribution.

Recall that $\mathcal{D}_t$ represents the *serving* data that is fed into ML models at *prediction* time. In the model (re)training step, if $\mathcal{D}_t$ is known, we can directly use $\mathcal{D}_t$ to adapt source data $\mathcal{D}_s$. Nevertheless, in practice, $\mathcal{D}_t$ is usually unseen until the model is deployed to start prediction. Thus, we introduce another notation $\mathcal{U}_t$. The scenario is that, although we might not know $\mathcal{D}_t$ when (re)training the model, it is usually not difficult to collect some *historical* data $\mathcal{U}_t$ in the same target domain of $\mathcal{D}_t$. Note that $\mathcal{U}_t$ does not have labels needed for training. Provided with $\mathcal{D}_s$ and $\mathcal{U}_t$, next we formalize the *adaptive data augmentation* problem.

Definition 1 (Adaptive Data Augmentation, ADA). *Given the labeled source data $\mathcal{D}_s = \{(\boldsymbol{x}_s^{(i)}, \boldsymbol{m}_s^{(i)}, y^{(i)})\}$ and the unlabeled data in the target $\mathcal{U}_t = \{(\boldsymbol{x}_t^{(i)}, \boldsymbol{m}_t^{(i)})\}$, the goal of adaptive data augmentation is to adapt $\mathcal{D}_s$ with regard to $\mathcal{U}_t$ and generate an augmented data $\tilde{\mathcal{D}}$, where a classifier f^+ trained on $\tilde{\mathcal{D}}$ would have low classification errors on the target data $\mathcal{D}_t$, i.e.,*

$$f^+ = \arg_f \min R_{\mathcal{D}_t}(f) = \arg_f \min Pr_{(\boldsymbol{x}_t, y) \in \mathcal{D}_t}[f(\boldsymbol{x}_t) \neq y] \quad (2)$$

Example 2. *Consider Figure 1. We want to train a classifier f that predicts whether a patient has cardiovascular disease (i.e.,* cardio = 1*). Specifically, source data $\mathcal{D}_s$, which could be curated by some research group, contains missing values on attributes* smoke *and* alcohol*. In contrast, target data is collected by clinics, and thus has a more sophisticated missing pattern that young patients tend to have more incomplete examination results, e.g.,* chol *and* gluc*, compared with old ones. Due to the "shift" of missing patterns, classifier f may not perform well on $\mathcal{D}_t$. To address this difficulty, we first adapt $\mathcal{D}_s$ with regard to historical unlabeled data in target $\mathcal{U}_t$ to generate an augmented data $\tilde{\mathcal{D}}$. Then, we patch f to a new classifier f^+ such that f^+ has low classification errors on the unseen target data $\mathcal{D}_t$.*

Note that, in the ADA problem, we consider the case of "one source, one target". It is natural to extend the problem to a more practical "one source, multiple targets" scenario: classifier f trained on the source data is deployed to serve *multiple* test sets $\{\mathcal{D}_t^1, \ldots, \mathcal{D}_t^m\}$ and each $\mathcal{D}_t^i$ has a different noise pattern.

Example 3. *Consider our example shown in Figure 1. While a classifier f is trained on the well-curated data $\mathcal{D}_s$ by a research group, f could be used in multiple hospitals with different missing patterns. For example, some hospitals may have incomplete demographics information (e.g.,* age*) while others may have missing values in* smoke *and* alchol*, which may be caused by the difference in workflow and equipments in the hospitals.*

A straightforward solution to handle "one source, multiple target" is to apply the above ADA for multiple times and obtain multiple "patched" models $f_1^+, \ldots, f_m^+$, each being applied in its corresponding target dataset. However, this method would bring more maintenance costs on model training, tuning and deployment. Thus, we study a more appealing alternative to retrain f to a single classifier f^+ for multiple target datasets. To this end, we define the adaptive data augmentation problem with multiple target datasets as follows.

Definition 2 (Multi-Target Augmentation, Multi-ADA). *Given a labeled source data $\mathcal{D}_s$ and a collection of unlabeled target datasets $\{\mathcal{U}_t^1, \mathcal{U}_t^2, \ldots, \mathcal{U}_t^m\}$, Multi-ADA generates an augmented dataset $\tilde{\mathcal{D}} = \tilde{\mathcal{D}}^1 \cup \tilde{\mathcal{D}}^2 \cup \ldots \cup \tilde{\mathcal{D}}^m$, where $\tilde{\mathcal{D}}^i$ is generated by adapting $\mathcal{D}_s$ with regard to $\mathcal{U}_t^i$. The objective is to train a classifier f^+ on $\tilde{\mathcal{D}}$ and minimize the classification errors on the unseen target data $\mathcal{D}_t = \mathcal{D}_t^1 \cup \ldots \cup \mathcal{D}_t^m$.*

We will show that one classifier f^+ learned on the augmented data with consistency regulation techniques in Section 5 can also achieve superior performance, compared with multiple classifiers.

Generative Adversarial Networks (GAN). GAN [18] is a generative model that tries to create *fake* data that resembles *real* data. GAN achieves this by pairing a *generator* G, which learns to produce near-real data, with a *discriminator* D, which learns to distinguish real data from the fake data generated by the generator. The *generator* never sees any real data, but tries to fool the discriminator that it can generate real data. To approximate the real data distribution $p(x)$, G takes a variable sampled from a noise distribution (e.g., a Gaussian distribution) $p(z)$ as input and outputs a fake data $G(z)$. The objective of *discriminator* is to distinguish the fake data $G(z)$ from the real data x sampled from $p(x)$ and guides the generator to generate more realistic data.

3 SOLUTION OVERVIEW

Let's start simple by considering a binary classification problem that classifies data points to two groups, a *true*-group and a *false*-group. Figure 2 gives an overview of our proposed framework.

Input. We have a labeled source $\mathcal{D}_s$, where the true (resp. false) data points are in the area of "+", the green circles (resp. "−", the pink triangles).

Output. Here, the "output" refers to using f to predict on the test data $\mathcal{D}_t$. The model f may cause both false positives (two red triangles) and false negatives (two light green circles).

Our proposal is a framework for adaptive data augmentation and model retraining. Besides $\mathcal{D}_s$, it also takes as input an unlabeled target data $\mathcal{U}_t$. It then learns to augment $\mathcal{D}_s$ by learning from $\mathcal{U}_t$, where the augmented source will be used to retrain the model and then predict on $\mathcal{D}_t$.

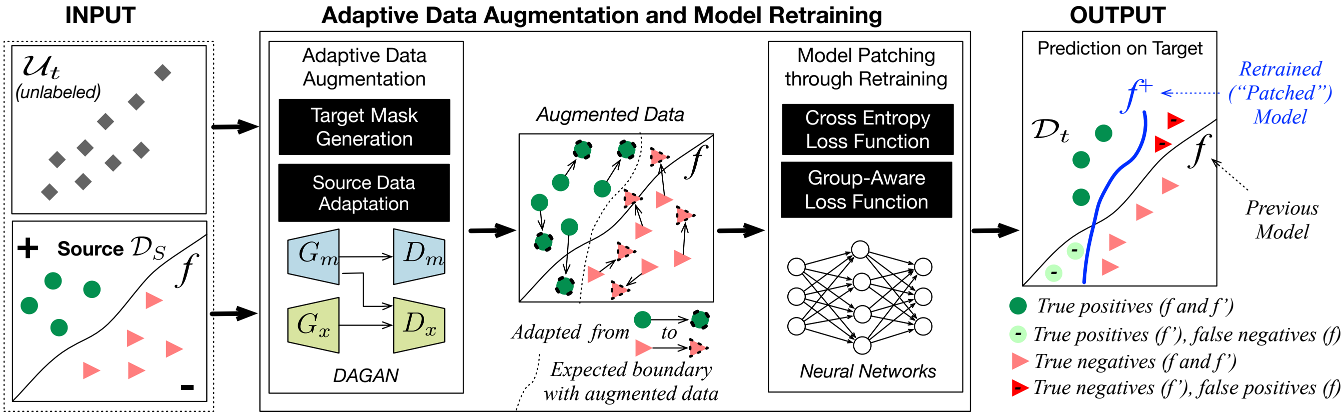

INPUT
Adaptive Data Augmentation and Model Retraining
OUTPUT
$\mathcal{U}_t$
(unlabeled)
+ Source $\mathcal{D}_S$
f
-
Adaptive Data Augmentation
Target Mask Generation
Source Data Adaptation
G_m
D_m
G_x
D_x
DAGAN
Augmented Data
f
Adapted from to
Expected boundary with augmented data
Model Patching through Retraining
Cross Entropy Loss Function
Group-Aware Loss Function
Neural Networks
Prediction on Target
$\mathcal{D}_t$
f^+
f
Retrained ("Patched") Model
Previous Model
● True positives (f and f')
● True positives (f'), false negatives (f)
▶ True negatives (f and f')
▶ True negatives (f'), false positives (f)

Figure 2: An overview of adaptive data augmentation.

Adaptive Data Augmentation (DAGAN). It also consists of two components: *target mask generation* and *source data adaptation.*

Target mask generation is to learn the noise pattern from the unlabeled target $\mathcal{U}_{\rm t}$. Source data adaptation is to augment the source data $\mathcal{D}_{\rm s}$ using the learned target (noise) mask.

To cope with the above two tasks, we use two GANs. One GAN, shown as $\langle G_m, D_m\rangle$, is to learn the missing value distribution of the target. The other GAN, shown as $\langle G_x, D_x\rangle$, is trained to learn the observed data distribution to augment the source.

As shown in Figure 2 "Augmented Data", each data point in the source data will be augmented by applying the learned target mask, which will result in a new data point. Moreover, it also shows that by doing the above data augmentation, if we retrain the model by considering these newly augmented data points, the model f, if being retrained, is expected to better fit the augmented data, as indicated by the dotted line.

Model Patching through Model Retraining. It will retrain the model f using the augmented and labeled source data. In particular, it also contains two components: *Cross Entropy Loss* for better modeling the adapted training data when there is only one target dirty dataset, and *Group-aware Loss* for retraining when there are multiple dirty target datasets (see more details in Section 5). The retrained model f^+ is shown as a bold line in the "output".

Output - Model Prediction on Test Data. After being retrained, f^+ can correctly predict the four wrong predictions made by the previous model f, that is, f^+ can correctly predict the two true positives (two pink circles) and two true negatives (two pink triangles) that f fails to predict. Note that, the missing data in target data $\mathcal{D}_{\rm t}$ will be imputed before being predicted.

4 GAN-BASED DATA AUGMENTATION

We first present the general architecture of DAGAN (Section 4.1), and describe the design details of *target mask generation* and *source data adaptation* (Section 4.2). We close this section by discussing the adversarial training algorithm of DAGAN (Section 4.3).

4.1 Architecture of DAGAN

The architecture of DAGAN is shown in Figure 3. It takes a sample of labeled source data $\mathcal{D}_{\rm s} = \{(\boldsymbol{x}_{\rm s}, \boldsymbol{m}_{\rm s})\}$ and a sample of unlabeled target data $\mathcal{U}_{\rm t} = \{(\boldsymbol{x}_{\rm t}, \boldsymbol{m}_{\rm t})\}$ as input. The goal of DAGAN is to "adapt" each tuple $(\boldsymbol{x}_{\rm s}, \boldsymbol{m}_{\rm s})$ in the source $\mathcal{D}_{\rm s}$ into a *synthetic tuple* $(\tilde{\boldsymbol{x}}_{\rm t}, \tilde{\boldsymbol{m}}_{\rm t})$. In particular, we want the synthetic tuple can follow target data distribution $p(\boldsymbol{x}_{\rm t}, \boldsymbol{m}_{\rm t})$. However, directly capturing $p(\boldsymbol{x}_{\rm t}, \boldsymbol{m}_{\rm t})$ is very challenging, as it may be a complex mixture of data distribution and noise distribution: the noise may be either independent (i.e., MCAR) or dependent (i.e., MAR and MNAR) of the data. Thus, we first deduce $p(\boldsymbol{x}_{\rm t}, \boldsymbol{m}_{\rm t})$ as

$$p(\boldsymbol{x}_{\rm t}, \boldsymbol{m}_{\rm t}) = p(\boldsymbol{m}_{\rm t}|\boldsymbol{x}_{\rm t}) \cdot p(\boldsymbol{x}_{\rm t}), \tag{3}$$

where $\boldsymbol{x}_{\rm t}$ is the observed data in the target. Based on this, we propose the following two essential tasks in DAGAN.

(1) Data distribution preservation: The first task aims to learn observed data distribution p($\boldsymbol{x}_{\rm t}$) that ensures that the fake data should be "similar" to the original source data. Formally, the synthetic tuple $\tilde{\boldsymbol{x}}_{\rm t}$ should preserve the data distribution. For example, we do not want the adapted data to contain tuples that violates the potential correlation between features and the `cardio` outcome.

(2) Missing pattern adaptation: This task aims to learn the underling missing pattern in the target. For example, we want G_m to learn that the target data as shown in Figure 1 has a missing pattern that young people (i.e., age ≤ 40) tend to have missing values on attributes `chol` and `gluc`. To this end, it aims to recover $p(\boldsymbol{m}_{\rm t}|\boldsymbol{x}_{\rm t})$ to ensure that the fake data should be adapted to have the synthetic mask $\tilde{\boldsymbol{m}}_{\rm t}$ that approximates the target missing pattern.

To achieve the aforementioned two properties, DAGAN introduces two generator-discriminator pairs $\langle G_m, D_m\rangle$ and $\langle G_x, D_x\rangle$, which are designed for masks and data respectively. The red line represents the process of G_m and D_m, where solid and dotted lines respectively represent the *forward* and *backward* passes. The green line represents the process of G_m, G_x and D_x. We present an overview of the two generator-discriminator pairs as follows.

Target Mask Generation. To enable our classifier to be familiar with the target data, we explicitly model the missing data process using a target mask generator. Since the target masks can be observed from $\mathcal{U}_{\rm t}$, we use a GAN model to estimate the target mask distribution. The key challenge here is that the missing pattern, i.e., mask distribution, may be dependent on the observed data. To address the challenge, as shown in the top of Figure 3, we utilize

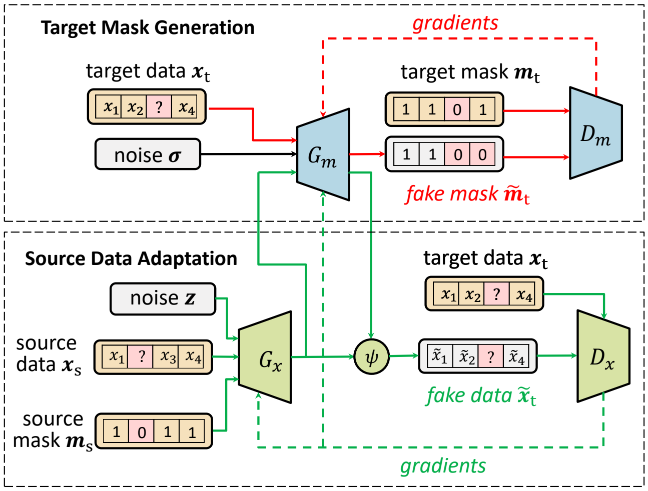

Target Mask Generation
gradients
target data $\boldsymbol{x}_\mathrm{t}$
x_1 x_2 ? x_4
noise $\boldsymbol{\sigma}$
G_m
target mask $\boldsymbol{m}_\mathrm{t}$
1 1 0 1
1 1 0 0
D_m
fake mask $\tilde{\boldsymbol{m}}_\mathrm{t}$
Source Data Adaptation
target data $\boldsymbol{x}_\mathrm{t}$
x_1 x_2 ? x_4
noise $\boldsymbol{z}$
source data $\boldsymbol{x}_\mathrm{s}$
x_1 ? x_3 x_4
G_x
ψ
$\tilde{x}_1$ $\tilde{x}_2$? $\tilde{x}_4$
D_x
fake data $\tilde{\boldsymbol{x}}_\mathrm{t}$
source mask $\boldsymbol{m}_\mathrm{s}$
1 0 1 1
gradients

Figure 3: The architecture of DAGAN.

two players, i.e., a mask generator G_m and a mask discriminator D_m. We use *conditional GAN* [31] in this phase. Given an observed data $\boldsymbol{x}_\mathrm{t}$ as condition, mask generator G_m transforms a random noise $\boldsymbol{\sigma}$ and to a fake mask $\tilde{\boldsymbol{m}}_\mathrm{t}$ corresponding to $\boldsymbol{x}_\mathrm{t}$. On the other hand, mask discriminator D_m is learned to distinguish $\tilde{\boldsymbol{m}}_\mathrm{t}$ from a real target mask $\boldsymbol{m}_\mathrm{t}$ from $\mathcal{U}_\mathrm{t}$, and guides G_m to generate near-real target masks.

Source Data Adaptation. To augment source data $\mathcal{D}_\mathrm{s}$, we train an adaptive data generator that "translates" each source tuple $(\boldsymbol{x}_\mathrm{s}, \boldsymbol{m}_\mathrm{s})$ into a fake target tuple $(\tilde{\boldsymbol{x}}_\mathrm{t}, \tilde{\boldsymbol{m}}_\mathrm{t})$, which is then compared with a real target tuple sampled from $\mathcal{U}_\mathrm{t}$. Same as the conditional GAN described above, it consists of a data generator G_x and a data discriminator D_x. Data generator G_x takes source data $\boldsymbol{x}_\mathrm{s}$ and source mask $\boldsymbol{m}_\mathrm{s}$ as *conditions* to transform a random noise $\boldsymbol{z}$ into a fake data $\tilde{\boldsymbol{x}}_\mathrm{t}$, which is then masked by the generated mask $\tilde{\boldsymbol{m}}_\mathrm{t}$ via function ψ in Equation (1). Discriminator D_x takes a real target data $\boldsymbol{x}_\mathrm{t}$ and the generated target data $\tilde{\boldsymbol{x}}_\mathrm{t}$ as input, and predicts whether $\tilde{\boldsymbol{x}}_\mathrm{t}$ is real or fake. By utilizing the adversarial training process, data generator G_x is improved to output fake tuples to fool D_x.

Example 4. *After the adversarial training process of DAGAN, we utilize the optimized generators G_m and G_x for adaptive data augmentation, as illustrated in Figure 4. More specifically, we first use mask generator G_m to output generated masks that explicitly model the missing pattern of the unlabeled target data $\mathcal{U}_\mathrm{t}$. Then, we feed the labeled source data to data generator G_x, and apply the ψ function to the output of G_x with generated masks to obtain the generated data where each source tuple is translated into a fake tuple that preserves source data distribution and has target missing patterns.*

4.2 Design Details of DAGAN

This section elaborates the two generator-discriminator pairs $\langle G_m, D_m\rangle$ and $\langle G_x, D_x\rangle$, which are respectively designed for target mask generation and source data adaptation.

Target Mask Generation. Recall that the difficulty of target mask generation is that the missing pattern, i.e., mask distribution, may be dependent on the observed data, given complicated missing mechanisms. In our previous example as shown in Figure 1, in the target data, the younger patients tend to have missing values in medical examinations. Thus, we feed this information to our mask generator G_m and would like G_m to be capable of learning such distribution of missing values. To achieve this objective, we implement mask generator G_m by improving the simple form of G_m in Figure 3 and introducing conditional GAN. In particular, mask generator $G_m(\boldsymbol{\sigma}, \boldsymbol{x})$ takes as input a random noise $\boldsymbol{\sigma} \in \mathbb{R}^\sigma$ and a sampled tuple to generate synthetic mask samples $G_m(\boldsymbol{\sigma}, \boldsymbol{x}) \in \mathbb{R}^N$. On the other hand, mask discriminator D_m distinguishes the generated masks by comparing the masks with real masks sampled from the target data. Specifically, $D_m(\boldsymbol{m}, \boldsymbol{x})$ takes a generated/real target mask $\boldsymbol{m}$ and the tuple $\boldsymbol{x}$ as input and determines the probability that $\boldsymbol{m}$ is sampled from $p(\boldsymbol{m})$, i.e., not generated from G_m. To train G_m and D_m, we follow Wasserstein GAN (WGAN) [7] and optimize them adversarially using the following loss function,

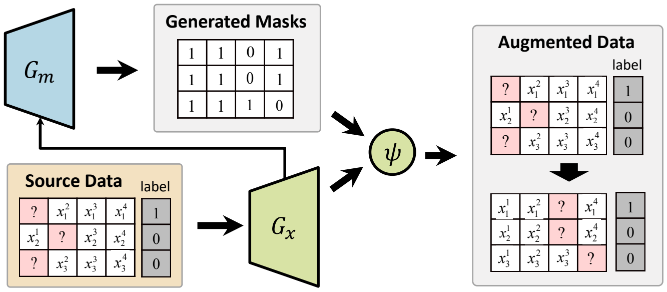


Figure 4: Adaptive data augmentation using DAGAN.

$$\begin{aligned}\mathcal{L}_m(D_m, G_m) = \ &\mathbb{E}_{(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})\sim\mathcal{U}_\mathrm{t}}[D_m(\boldsymbol{m}_\mathrm{t}, \boldsymbol{x}_\mathrm{t})] \\ &- \mathbb{E}_{\boldsymbol{\sigma}\sim p(\boldsymbol{\sigma}),(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})\sim\mathcal{U}_\mathrm{t}}[D_m(G_m(\boldsymbol{\sigma}, \boldsymbol{x}_\mathrm{t}), \boldsymbol{x}_\mathrm{t})]\end{aligned} \tag{4}$$

Source Data Adaptation. The objective of source data adaptation is to adapt each tuple $(\boldsymbol{x}_\mathrm{s}, \boldsymbol{m}_\mathrm{s})$ in $\mathcal{D}_\mathrm{s}$ into a *fake tuple* $(\tilde{\boldsymbol{x}}_\mathrm{t}, \tilde{\boldsymbol{m}}_\mathrm{t})$ that satisfies the aforementioned label preserving and missing pattern adaptive properties. To this end, we adopt a conditional GAN architecture [31] that consists of a *data generator* G_x and a *data discriminator* D_x: G_x takes source tuples as condition and adapt the tuples to mimic the missing pattern of target data $\mathcal{U}_\mathrm{t}$, while D_x distinguishes the adapted tuples with the real ones.

Formally, *data generator* $G_x(\boldsymbol{z}, \boldsymbol{x}_\mathrm{s}, \boldsymbol{m}_\mathrm{s})$ takes as input a random noise $\boldsymbol{z} \in \mathbb{R}^z$ and a source tuple $(\boldsymbol{x}_\mathrm{s}, \boldsymbol{m}_\mathrm{s})$. It first generates an intermediate complete tuple $\tilde{\boldsymbol{x}}$, and then masks $\tilde{\boldsymbol{x}}$ by applying the synthetic mask $\tilde{\boldsymbol{m}}_\mathrm{t}$ generated by our mask generator G_m, i.e.,

$$\begin{aligned}\tilde{\boldsymbol{x}}_\mathrm{t} &= \psi(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{m}}_\mathrm{t}) \\ &= G_x(\boldsymbol{z}, \boldsymbol{x}_\mathrm{s}, \boldsymbol{m}_\mathrm{s}) \odot \tilde{\boldsymbol{m}}_\mathrm{t} + \mathsf{NA} \cdot (1 - \tilde{\boldsymbol{m}}_\mathrm{t})\end{aligned} \tag{5}$$

Data discriminator D_x is designed to distinguish the synthetic data with real data sampled from target data. We simultaneously consider the above G_x and D_x in the training process. First, we sample source data $\boldsymbol{x}_\mathrm{s}$ and obtain an intermediate tuple $\tilde{\boldsymbol{x}}$ using G_x. Next, we take $\tilde{\boldsymbol{x}}$ as the input of G_m to generate the corresponding mask $\tilde{\boldsymbol{m}}_\mathrm{t}$, and then we synthesize a fake target tuple $\tilde{\boldsymbol{x}}_\mathrm{t}$ by masking $\tilde{\boldsymbol{x}}$ with $\tilde{\boldsymbol{m}}_\mathrm{t}$. Finally, the discriminator D_x compares the fake tuple $\tilde{\boldsymbol{x}}_\mathrm{t}$ with a real observed one sampled from target data $\mathcal{U}_\mathrm{t}$.

We can optimize G_m jointly with G_x and D_x according to the loss function,

$$\begin{aligned}\mathcal{L}_x(D_x,G_x,G_m) &= \mathbb{E}_{(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})\sim\mathcal{U}_\mathrm{t}}[D_x(\boldsymbol{x}_\mathrm{t})] \\ &- \mathbb{E}_{\boldsymbol{\sigma}\sim p(\boldsymbol{\sigma}),\boldsymbol{z}\sim p(\boldsymbol{z}),(\boldsymbol{x}_\mathrm{s},\boldsymbol{m}_\mathrm{s})\sim\mathcal{D}_\mathrm{s}}[D_x(\psi(\tilde{\boldsymbol{x}},G_m(\boldsymbol{\sigma},\tilde{\boldsymbol{x}})))], \tilde{\boldsymbol{x}}=G_x(\boldsymbol{z},\boldsymbol{x}_\mathrm{s},\boldsymbol{m}_\mathrm{s})\end{aligned} \tag{6}$$

Note that the losses in Definitions (4) and (6) use the loss function of Wasserstein GAN [7]. Overall, we combine the two objectives in our adversarial training process, i.e.,

$$\min_{G_x}\max_{D_x}\mathcal{L}_x(D_x,G_x,G_m) \tag{7}$$

$$\min_{G_m}\max_{D_m}\mathcal{L}_m(D_m,G_m)+\mathcal{L}_x(D_x,G_x,G_m) \tag{8}$$

Neural Network Architectures. The works [14, 15] have shown that LSTM, a representative variant of RNN, can achieve superior performance when used in generators for relational data. The basic idea is to formalize record synthesis as a *sequence generation* process: a record $\boldsymbol{x}$ is modeled as a sequence and each element of the sequence is an attribute $\boldsymbol{x}_j$. Then we use LSTM to generate $\boldsymbol{x}$ at multiple timesteps, where the j-th timestep is used to generate $\boldsymbol{x}_j$.

Formally, let $\boldsymbol{h}^j$ and $\boldsymbol{f}^j$ respectively denote the hidden state and output of the LSTM at the j-th timestep. Then, we have

$$\begin{aligned}\boldsymbol{h}^{j+1} &= \mathtt{lstm}(z,\boldsymbol{f}^j,\boldsymbol{h}^j),\\ \boldsymbol{f}^{j+1} &= \mathtt{tanh}(\mathtt{FC}(\boldsymbol{h}^{j+1})),\end{aligned} \tag{9}$$

where $\boldsymbol{h}^0$ and $\boldsymbol{f}^0$ are initialized with random values and FC denotes fully-connected layer. Next, we compute attribute $\boldsymbol{x}_j$ using an output layer. For discriminators, we follow the original GAN [18] to use multi-layer perceptron (MLP). More details of the neural network architectures can be referred to our technical report [15].

4.3 Training Algorithm of DAGAN

Based on the above objectives, we train DAGAN using the process shown in Algorithm 1. In each training iteration, we follow Wasserstein GAN [7] to first update the parameters of discriminators D_m and D_x and then train the generators G_m and G_x.

In the training process, we sample a real tuple $(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})$ from target data distribution and generate a fake tuple $(\tilde{\boldsymbol{x}}_\mathrm{t},\tilde{\boldsymbol{m}}_\mathrm{t})$ using G_x and G_m. We take these tuples as the input of D_x and D_m then update the parameters of DAGAN by maximizing or minimizing the output of D_x and D_m.

For the mask discriminator D_m, we treat it as a normal discriminator of conditional GAN. D_m maximizes the output score of the input $(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})$ sampled from the distribution of target data $p(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})$ and minimizes the output score of the input $(\boldsymbol{x}_\mathrm{t},\tilde{\boldsymbol{m}}_\mathrm{t})$ whose mask $\tilde{\boldsymbol{m}}_\mathrm{t}$ is generated by the mask generator G_m. The data discriminator D_x will separately take a real target tuple $\boldsymbol{x}_\mathrm{t}$ and a synthetic target tuple $\tilde{\boldsymbol{x}}_\mathrm{t}$ generated by G_x as inputs and try to maximize the difference between the output scores of them.

In this process, we first fix the parameters of G_x and G_m to train the D_x (lines 3–6) and D_m (lines 7–11), then fix the parameters of D_x and D_m to train the G_x and G_m (lines 12–20).

Algorithm 1: The Pseudo-code of DAGAN Training

```
   Input: parameters $\theta_{g_x}$, $\theta_{d_x}$ of $G_x$ and $D_x$; parameters $\theta_{g_m}$, $\theta_{d_m}$ of
          $G_m$ and $D_m$; $T$: number of training iterations
   Output: The learned models
1  Initialize parameters $\theta_{g_x}$,$\theta_{d_x}$ for $G_x$,$D_x$ and $\theta_{g_m}$,$\theta_{d_m}$ for $G_m$,$D_m$
2  for training iteration $t_1 = 1, 2, \ldots, T_g$ do
       /* Training discriminator $D_m$ */
3      Randomly sample $\boldsymbol{\sigma}$ and $(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})$ from noise prior $p(\boldsymbol{\sigma})$ and
       target data $p(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})$;
4      Compute $L_{d_m}$ using $G_m$ and $D_m$ with Eq (4);
5      Compute the gradients $\nabla_{\theta_{d_m}} L_{d_m}$;
6      Update $\theta_{d_m}$ using stochastic gradient descent.
       /* Training discriminator $D_x$ */
7      Randomly sample $\boldsymbol{\sigma}$ and $\boldsymbol{z}$ from noise prior $p(\boldsymbol{\sigma})$ and $p(\boldsymbol{z})$;
8      Randomly sample $(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})$ and $(\boldsymbol{x}_\mathrm{s},\boldsymbol{m}_\mathrm{s})$ from target data
       $p(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})$ and source data $p(\boldsymbol{x}_\mathrm{s},\boldsymbol{m}_\mathrm{s})$;
9      Compute $L_{d_x}$ using $G_x$, $G_m$ and $D_x$ with Eq (6);
10     Compute the gradients $\nabla_{\theta_{d_x}} L_{d_x}$;
11     Update $\theta_{d_x}$ using stochastic gradient descent.
       /* Training generators $G_m$ and $G_x$ */
12     Randomly sample $\boldsymbol{\sigma}$ and $\boldsymbol{z}$ from noise prior $p(\boldsymbol{\sigma})$ and $p(\boldsymbol{z})$;
13     Randomly sample $(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})$ and $(\boldsymbol{x}_\mathrm{s},\boldsymbol{m}_\mathrm{s})$ from target data
       $p(\boldsymbol{x}_\mathrm{t},\boldsymbol{m}_\mathrm{t})$ and source data $p(\boldsymbol{x}_\mathrm{s},\boldsymbol{m}_\mathrm{s})$;
14     Compute $L_{g_x}$ using $G_x$, $G_m$ and $D_x$ with Eq (6);
15     Compute gradients $\nabla_{\theta_{g_x}} L_{g_x}$;
16     Update $\theta_{g_x}$ using stochastic gradient descent.
17     Compute $L_m$ using $G_m$ and $D_m$ with Eq (4);
18     $L_{g_m} \leftarrow L_{g_x} + L_m$;
19     Compute gradients $\nabla_{\theta_{g_m}} L_{g_m}$;
20     Update $\theta_{g_m}$ using stochastic gradient descent.
21 return $G_x$, $D_x$, $G_m$, $D_m$
```

5 MODEL RETRAINING

5.1 Model Retraining for ADA

Model retraining is simple for the basic ADA problem with single target dataset. Specifically, we denote augmented data as $\tilde{D} = \{\boldsymbol{x}_i, y_i\}$, which is generated by adapting source data $\mathcal{D}_\mathrm{s}$ into unlabeled target data $\mathcal{U}_\mathrm{t}$. We first apply a *simple imputation* method for $\tilde{D}$, i.e., imputing numerical attributes with the mean of observed values, and replacing missing values of each categorical attribute with a special category (such as zero). Note that such simple imputation method is much more efficient than the sophisticated imputation solutions [36, 43, 50]. After that, we train our classifier f on the augmented data using the empirical risk minimization (ERM) [44] criterion, i.e., optimizing the following loss function,

$$L(\theta) = \mathbb{E}_{(\boldsymbol{x}_i,y_i)\in\tilde{\mathcal{D}}}[\mathcal{L}(f_\theta(\boldsymbol{x}_i),y_i)], \tag{10}$$

where $\mathcal{L}$ is the loss between a predicted label $f_\theta(\boldsymbol{x}_i)$ and ground-truth y_i. We use the widely-adopted *cross entropy* function to implement $\mathcal{L}$ in our experiments.

5.2 Model Retraining for Multi-ADA

A straightforward approach to model retraining for Multi-ADA is to use the ERM-based method [44] from basic ADA. More specifically, consider the augmented data $\tilde{\mathcal{D}} = \tilde{\mathcal{D}}^1 \cup \ldots \cup \tilde{\mathcal{D}}^m$, where each $\tilde{\mathcal{D}}^i$ is generated by adapting source data $\mathcal{D}_\mathrm{s}$ into unlabeled data $\mathcal{U}^i_\mathrm{t}$. We do not differentiate the augmented data corresponding to various targets, and simply train our classifier f^+ on $\tilde{\mathcal{D}}$ using the loss function in Equation (10). Finally, we apply f^+ on various

Algorithm 2: Model Retraining for Multi-ADA using GDRO

```
Input: $b$: batch size; $\alpha$: learning rate; $T$: number of training iterations
Output: $f_\theta$: classifier;
Initialize parameters $\theta^{(0)}$ for $f_\theta$
for training iteration $t_1 = 1, 2, \dots, T$ do
    Sample $b$ samples $\{\boldsymbol{x}_m^{(i)}, \boldsymbol{y}_m^{(i)}\}_{i=1}^{b}$ from each subgroup $\tilde{\mathcal{D}}^m$
    $Loss \leftarrow -Inf$
    for samples from each subgroup do
        $m\_loss \leftarrow \mathcal{L}(\boldsymbol{y}_m, f_\theta(\boldsymbol{x}_m))$
        if $m\_loss > Loss$ then
            $Loss \leftarrow m\_loss$
    $g \leftarrow \nabla_\theta Loss$
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \alpha \cdot \mathsf{adam}(\theta^{(t-1)}, g)$
return $f_\theta$
```

Table 1: Dataset statistics: #Rec (records), #N (numerical attributes), #C (categorical attributes), and #L (unique labels).

Dataset	Area	#Rec	#N	#C	#L	Label Distribution
Ipums	Social	16329	16	43	7	65:9:11:5:6:3:1
Okcupid	Social	50789	3	13	3	7 : 2 : 1
Welfare	Financial	20309	5	0	2	1.0 : 1.2
EyeState	Health	14977	14	0	2	1.0 : 1.3
Adult	Social	13567	6	8	2	1.0 : 3.0

target datasets $\mathcal{D}_{\mathrm{t}}^1, \dots, \mathcal{D}_{\mathrm{t}}^m$. For ease of presentation, we call each $\mathcal{D}_{\mathrm{t}}^i$ as a *subgroup* of the total target data $\mathcal{D}_{\mathrm{t}}$.

Challenge of Subgroup Performance Gap. However, we have a supervising observation from our experiments that the above approach has inferior performance compared with multiple classifiers, each of which is retrained on $\mathcal{D}_{\mathrm{t}}^i$. A deeper analysis shows that f^+ encounters a difficulty of *subgroup performance gap*. More specifically, as observed from Figure 11 (for details please refer to Section 6.6), the losses of different test subgroups are very *diverse*: with the increase of training epochs, although losses of some subgroups are optimized, there still exist many subgroups with large losses. The main reason is that the ERM criterion ignores the difference among subgroups, and tends to optimize the losses of "easy" subgroups with larger gradients. In contrast, for the "difficult" subgroups with smaller gradient, even when their losses are significant, they may not get chance to be optimized by the ERM approach.

Subgroup-aware Model Retraining. To address the above challenge, we adopt a simple yet effective optimization technique: group distributionally robust optimization (GDRO) [40]. Different from ERM which aims to optimize the overall loss, GDRO considers to optimize the subgroup with the *largest loss* iteratively. Algorithm 2 shows the pseudo code of GDRO. In each iteration, it finds the subgroup with the largest loss to optimize, i.e.,

$$L(\theta) = \max_m \mathbb{E}_{(\boldsymbol{x}_i, y_i) \in \tilde{\mathcal{D}}^m} [\mathcal{L}(f_\theta(\boldsymbol{x}_i), y_i)], \tag{11}$$

6 EXPERIMENTS

6.1 Setup

Datasets. We utilize both real-world datasets with (i) real missing values and (ii) datasets with injected missing values. The statistics of tested datasets are summarized in Table 1. Specifically, we use the following three real-world datasets.

1) Ipums is the Public Use Microdata Sample (PUMS) census data from the Los Angeles and Long Beach areas [3]. Following the common practice in ML, we partition the dataset into source and target based on the timestamp and train a classifier for predicting attribute MovedIn. In particular, we use the tuples in 1998 as the source and tuples in 1999 as the target. The source has 17 attributes with missing rates ranging from 1.8% to 71.0%, and the target has 18 attributes with missing rates ranging from 1.9% to 69.8%.

2) Okcupid contains user profile data for San Francisco OkCupid users [4]. It includes people within a 25 mile radius of San Francisco, who were online in 2011. We train a classifier to predict the job of a user from three categories, STEM, non-STEM and Student, where STEM stands for jobs in computer/hardware/software/science/tech/engineering. Like the Ipums dataset, we use the data with last online time before 2012-06-27 18:04 as the source data, and use the rest as the target data. The source has 13 attributes with missing rates ranging from 0.1% to 74.8%, and the target has 13 attributes with missing rates ranging from 0.04% to 80.1%.

3) Welfare contains financial statistics of 4180 foundations in China from 2005 to 2016 [5]. We train a classifier for predicting the welfare expense of each foundation in the future. We create the source data with tuples before 2011 and train the classifier on the source. Then, we deploy the classifier on the target data with tuples after 2011. The source has 2 attributes with missing rates 0.1% and 34.6%, and the target has 5 attributes with missing values with rates ranging from 0.1% to 15.1%.

Please find more details about the missing value of each attribute in our technical report [24].

Moreover, to comprehensively evaluate approaches under different missing value patterns, like the existing works [46, 50], we manually inject missing values to clean datasets. We use the following two representative ML datasets, which are downloaded from the UCI Machine Learning Repository.

4) EyeState contains records obtained from one continuous EEG (electroencephalogram) measurement with the Emotiv EEG Neuroheadset [2]. Specifically, each record has multiple attributes corresponding to various metrics measured by the Neuroheadset, and all the attributes in the dataset are *numerical*. Moreover, a label corresponding to the eye state is associated with each record, which is one of two possible values, 1 (eye-closed) and 0 (eye-open).

5) Adult contains personal information from the 1994 US census [1]. Each record in the dataset corresponds to a person with *mixed* data types, i.e., 8 categorical and 6 numerical attribute values. The attribute income is used as the label to predict whether the person has income larger than 50K per year (positive) or not (negative). Note that, on the full Adult dataset, baselines would incur high runtime cost. For example, MICE and MISF respectively use more than 2 hours and 10 hours to complete missing value imputation. In contrast, our DAGAN approach only takes 40 minutes for both GAN learning and data adaptation. To favor the baselines, we randomly sample a subset of records from the dataset.

Missing Value Injection. We consider the following settings where the source and the target have different missing patterns. (i) The NoOverlap setting considers missing values occur in different sets of attributes in source and target data. We apply a mechanism that injects missing values to the *first half* of attributes in the source, and injects missing values to the *last half* of attributes in the target. (ii) The `Overlap` setting injects missing values to the same set of attributes in the source and the target with *different* missing rates.

We consider various widely-recognized mechanisms for injecting missing values, namely MCAR, MAR and MNAR, in the aforementioned settings. MCAR is the simplest mechanism with a strong assumption that missing values are *independent* of the data. In contrast, MAR and MNAR are more realistic as they consider that missing values depend on the underlying data.

1) *Missing Completely at Random (MCAR)*: Given a missing rate p and a set of attributes in a dataset, the MCAR mechanism changes every attribute of each tuple in the dataset to the missing value `NA` completely at random with probability p.

2) *Missing at Random (MAR).* Different from MCAR, MAR considers that the missing is not completely at random, but is dependent on the values of some observed attributes. Specifically, we follow the method in [46] that first selects a numerical attribute, denoted as A_R, and sorts the tuples in ascending order of A_R. Next, we inject missing values for each of the other attributes A: given a missing rate p, we randomly select a continuous range of tuples with size $p \cdot M$ (M is the number of tuples). For each selected tuple, we replace the corresponding value of A with missing value `NA`.

3) *Missing Not at Random (MNAR).* MNAR considers data missing dependent on both observed and unobserved attributes. We also follow the method in [46]. For each attribute A, we sort all the tuples in ascending order of A and then choose the largest or smallest p proportion of the tuples to inject missing values.

Evaluation Framework. We implement the DAGAN framework as shown in Figure 3, using PyTorch. All the code and datasets in our experiments are public at Github[1]. To evaluate the performance of DAGAN, we split each dataset $\mathcal{D}$ into source data $\mathcal{D}_\mathrm{s}$, unlabeled data in target $\mathcal{U}_\mathrm{t}$ and unlabeled test data $\mathcal{D}_\mathrm{t}$ with the ratio of 4:5:1. Then, we apply the missing value injection methods, which are described previously, to the datasets $\mathcal{D}_\mathrm{s}, \mathcal{U}_\mathrm{t}$ and $\mathcal{D}_\mathrm{t}$. Next, we train DAGAN by using the source data $\mathcal{D}_\mathrm{s}$ and unlabeled data $\mathcal{U}_\mathrm{t}$ to obtain the optimized parameters of discriminators and generators as follows. We first perform hyper-parameter search, which will be described later, to determine the hyper-parameters of the model. Then, we run a training algorithm for parameter optimization and adapt the source data $\mathcal{D}_\mathrm{s}$ to the augmented data $\tilde{\mathcal{D}}$. Then, we train our classifier f^+ on $\tilde{\mathcal{D}}$ and apply f^+ to $\mathcal{D}_\mathrm{t}$. We use F1 score, which is the harmonic average of precision and recall, as the evaluation metric for the classifier. In particular, for binary classifier, we measure the F1 score of the positive label. For multi-label classifier, we measure the weighted F1 across different labels.

Baseline Approaches. We compare our GAN-based framework DAGAN with state-of-the-art data imputation methods. For a fair comparison, we first use both source data $\mathcal{D}_\mathrm{s}$ and unlabeled target data $\mathcal{U}_\mathrm{t}$ to train a data imputation method, and use test data $\mathcal{D}_\mathrm{t}$ to train its imputation method. Then, we use the corresponding methods to impute $\mathcal{D}_\mathrm{s}$ and $\mathcal{D}_\mathrm{t}$ respectively. After that, we train our classifier on the imputed $\mathcal{D}_\mathrm{s}$ and evaluate the classifier's performance on the imputed $\mathcal{D}_\mathrm{t}$. We consider the following three state-of-the-art data imputation methods.

(1) **Generative Adversarial Imputation Nets (GAIN)** [50] is a recently proposed generative method for data imputation. It first trains a GAN model to approximate the data distribution, and then uses the model to impute missing values. We download the source code of GAIN from https://github.com/jsyoon0823/GAIN and use the default parameters provided by the authors.

(2) **MissForest (MISF)** [43] is a nonparametric, discriminative imputation method based on random forest. It first uses the observed data to train a random forest classifier, and then uses the classifier to predict the missing values. We use their original implementation of MISF from https://github.com/stekhoven/missForest.

(3) **Multiple Imputation with Chained Equations (MICE)** [36] is one of the most recognized imputation methods in statistics and data science. It is also a discriminative imputation method that predicts missing values from observed data. Different from MISF, it is a parametric method that uses classification and regression model for imputation. We use their original implementation of MICE from https://github.com/amices/mice.

Classification Model. For the classification model, we use the representative deep learning model, multi-layer perceptron (MLP) with two hidden layers with neuron numbers 100 and 50. We use `ReLu` and `sigmoid` as activation functions for hidden layer and output layer respectively. During model training, we use *binary cross-entropy* as the loss function, and use `Adam` [19] as the optimizer with the default learning rate 0.01. Note that the selection of classification models is orthogonal to our paper, and we will discuss more sophisticated classification models in the future work.

Hyper-parameter Search. Hyper parameter search is crucial for deep learning models. For training the GAN-based model in DAGAN, we adopt the method in a recent empirical study for GAN models [26] for hyper-parameter search. We first generate a set of candidate hyper-parameter settings, and train the models for several times, each of which corresponds to a hyper-parameter setting. Then, we select the models with the best performance on the validation set. We also perform hyper-parameter search for our classifier as follows. We first sample a subset from the training dataset as validation set, and then train a classifier f on the remaining data. We divide the training iterations evenly into 10 epochs and evaluate the performance of the model snapshot after each epoch on the validation data. We select the model snapshot with the best performance. In particular, we use full batch for all experiments and set 2000 as the number of maximum iterations, so the only parameter we need to tune is the coefficient of L2 normalization. The candidate coefficients set is $\{0, 0.001, 0.01, 0.1\}$. For each candidate coefficient, we use it to train a model and evaluate it on validation data. We select the coefficient with the best performance.

All experiments are conducted on a Ubuntu 16.04 server with 62TB disk, 40 CPU cores (Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz), one GPU (NVIDIA TITAN V) and 512GB memory, and the version of Python is 3.6.12.

[1] https://github.com/ruc-datalab/dagan

Table 2: Comparison on Real-World Datasets on F1 scores.

Dataset	GAIN	MISF	MICE	DAGAN
Ipums	0.709	-	0.723	**0.768**
Okcupid	0.597	0.649	0.663	**0.681**
Welfare	0.477	0.536	0.579	**0.652**

Table 3: Evaluating Effect of Data Adaptation on F1 scores.

(a) EyeState Dataset

Missing Rate	MCAR		MAR		MNAR	
	AdaSrc	ClnSrc	AdaSrc	ClnSrc	AdaSrc	ClnSrc
0.2	0.647	**0.655**	0.665	**0.695**	0.668	**0.681**
0.3	**0.676**	0.650	0.641	**0.680**	0.699	**0.716**
0.4	**0.644**	0.625	0.622	**0.681**	0.686	**0.698**
0.5	**0.631**	0.563	**0.660**	0.544	0.665	**0.690**
0.6	**0.618**	0.554	**0.638**	0.602	**0.666**	0.622
0.7	**0.570**	0.485	**0.669**	0.496	**0.620**	0.610
0.8	**0.576**	0.494	**0.626**	0.538	**0.630**	0.585

(b) Adult Dataset

Missing Rate	MCAR		MAR		MNAR	
	AdaSrc	ClnSrc	AdaSrc	ClnSrc	AdaSrc	ClnSrc
0.2	0.615	**0.616**	**0.621**	0.610	**0.664**	0.593
0.3	**0.629**	0.569	**0.621**	0.563	**0.623**	0.578
0.4	**0.632**	0.581	**0.643**	0.585	**0.617**	0.593
0.5	**0.623**	0.539	**0.603**	0.590	**0.576**	0.546
0.6	0.579	**0.582**	0.616	**0.645**	**0.600**	0.592
0.7	**0.589**	0.464	**0.593**	0.512	**0.595**	0.492
0.8	**0.596**	0.471	**0.614**	0.561	**0.588**	0.570

Table 4: Evaluating Neural Network Design on F1.

(a) EyeState Dataset

Missing	Gen	Missing Rate						
		0.2	0.3	0.4	0.5	0.6	0.7	0.8
MCAR	MLP	**0.68**	**0.63**	0.54	0.57	0.54	0.52	0.43
	LSTM	0.65	0.62	**0.61**	**0.60**	**0.61**	**0.56**	**0.57**
MAR	MLP	**0.65**	0.61	0.59	0.57	0.59	0.53	0.55
	LSTM	0.64	**0.65**	**0.63**	**0.62**	**0.60**	**0.60**	**0.59**
MNAR	MLP	0.69	0.68	0.64	**0.67**	0.63	**0.61**	**0.64**
	LSTM	**0.70**	0.68	**0.66**	0.62	**0.67**	0.59	0.61

(b) Adult Dataset

Missing	Gen	Missing Rate						
		0.2	0.3	0.4	0.5	0.6	0.7	0.8
MCAR	MLP	0.60	0.59	0.56	0.58	0.41	0.53	0.28
	LSTM	0.60	**0.61**	0.56	**0.59**	**0.58**	**0.57**	**0.60**
MAR	MLP	0.58	0.52	0.56	0.54	0.53	**0.56**	0.49
	LSTM	0.58	**0.61**	**0.58**	**0.55**	**0.57**	0.55	**0.59**
MNAR	MLP	**0.60**	0.57	0.54	0.58	**0.59**	0.55	0.53
	LSTM	0.59	0.57	**0.56**	**0.62**	0.58	**0.57**	**0.54**

6.2 Evaluation on Real-World Datasets

We first evaluate the performance of DAGAN on real-world datasets with missing values. Specifically, we adopt an imputation approach to respectively impute source and test data. To make a fair comparison, when imputing source data, we also leverage the unlabeled target data for training the imputation model. After obtaining the imputed source and test data, we train our classifier on the source data and evaluate performance of the classifier on the test data.

Table 2 summarizes the result. It clearly shows that DAGAN outperforms existing methods when handling noise shift between source and target on these real-world datasets. The performance improvement is attributed to our adaptive data augmentation approach, which tries to avoid the dataset shift between the source and the target data. Note that MISF fails on the Ipums dataset because its R-based implementation cannot handle categorical attributes with more than 53 categories, as noted in its instruction.

6.3 Effect of Adaptive Data Augmentation

To provide in-depth analysis on adaptive data augmentation, we consider the datasets with injected missing values. Specifically, we consider the NoOverlap setting as described in Section 6.1. For the source data, we fix the missing rate as 0.6 and use MCAR to inject missing values to the first half of attributes. For the target data, we vary missing rate p from the values in $\{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$, and only inject missing values to the last half of attributes. Note that we observe similar results on other missing injection settings. We omit the results due to space limit.

The key question we answer is whether an *adaptively augmented* source data is better than a *fully clean* source data, when the target data is dirty. To this end, we consider two alternatives. (1) ClnSrc uses a perfectly clean source data to train a classifier, and evaluates the classifier on the dirty test data. (2) AdaSrc first adapts the source data with the target noise pattern based on the injected missing pattern in the target data. Then, it trains a classifier on the adapted source data and evaluates the classifier on the test. Table 3 reports the experimental results under missing patterns MCAR, MAR and MNAR. One interesting observation is that, although ClnSrc achieves better performance than AdaSrc under small missing rates, AdaSrc outperforms ClnSrc with larger margin on F1 with the increase of missing rate. For example, consider the EyeState result in Table 3(a). Under the MCAR setting, only when missing rate $p = 0.2$, ClnSrc is slightly better than AdaSrc, but under other missing rates, AdaSrc achieves 3%-18% improvement on F1. Similarly, under the MAR and MNAR settings, AdaSrc outperforms ClnSrc for missing rates $p > 0.4$ and $p > 0.5$ respectively. On the Adult dataset, AdaSrc is much better and outperforms ClnSrc in most cases.

We analyze the experimental results as follows. ClnSrc cannot achieve satisfactory results mainly because the source and the target data have divergence on data distribution. Specifically, although the source data is fully clean, the imputed data in the target may not be the ground-truth. Subsequently, the classifier trained on the source may encounter *dataset shift* when serving the target. As validated in the ML community [9, 23, 42], dataset shift would severely affect the performance of ML models. In contrast, AdaSrc enables the source data to be adapted to target data distribution. The experimental results show that such adaptation is beneficial for ML training. The results also motivate us to design *target mask generation* in the architecture of DAGAN: we use target mask generation to learn the missing pattern of the target and reduce the distributional difference between source and target data.

6.4 Evaluation on Model Design of DAGAN

We have designed a GAN-based approach DAGAN to fulfill adaptive data augmentation. Next we examine the design choices in the architecture of DAGAN. We adopt the same missing injection setting used in the previous section. We focus on the neural network

(NN) design of the data generator G_x, and consider two widely-used neural network models, MLP and LSTM, which are presented in Section 4.2. Note that we do not evaluate Convolutional Neural Network (CNN) for data generator [10, 33], because the existing work has shown that CNN achieves inferior performance in relational data generation [14]. Moreover, we use MLP to implement mask generator G_m, because the process of mask generation is relatively simple. We also evaluate LSTM for mask generator and obtain similar results to that of MLP. As observed from Table 4, LSTM achieves better performance than MLP in most of the cases on our datasets. Similar results are also reported in a recent empirical study [14]. The main reason is that LSTM generates an attribute based on the "understanding" of previous attributes, and thus it would be capable of capturing attribute correlation. In the remainder of this section, we use LSTM as the default NN model for data generator G_x.

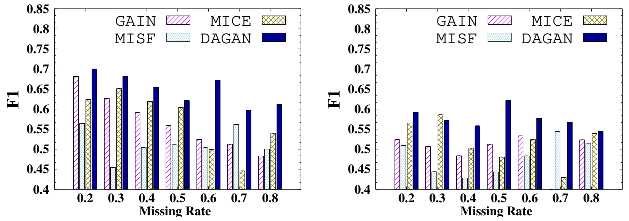
6.5 Robustness for Different Missing Patterns

This section evaluates the robustness of DAGAN for handling different missing value patterns, compared with the state-of-the-art data imputation approaches. To make a comprehensive comparison, we consider three different settings of missing value injection on the EyeState and Adult datasets.

Evaluation under NoOverlap noise injection setting. We evaluate robustness of approaches under the NoOverlap missing value injection setting. Note that in practice, in many machine learning and deep learning projects, high missing value rates are not rare. For example there are many tables in OpenML (https://www.openml.org) with high missing value rates. We provide statistics of some representatives of such tables in our technical report [24] due to space limit. Thus, this section first considers the scenarios of high missing rates. Specifically, we fix the missing rate as 0.6 to inject missing values to the first half of attributes in the source data, while varying missing rate p from the values in $\{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ to inject missing values to the last half of attributes in the target data.

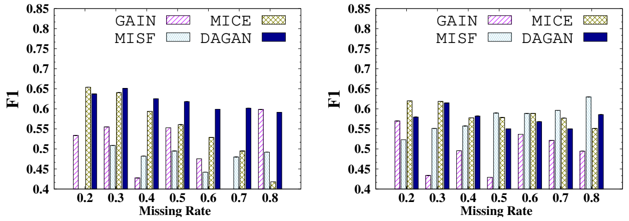
Figures 5 and 6 show the experimental results on the *difficult* scenarios where missing values are injected according to MNAR and MAR. In these scenarios, missing values are dependent on the underlying data, and thus the missing pattern is non-trivial to be identified. The result shows that DAGAN outperforms the other data imputation methods in most of the cases. For example, on the EyeState dataset and under the MAR setting, DAGAN respectively outperforms MICE, MISF and GAIN by 11.11%, 31.79% and 22.66% on average. Moreover, DAGAN also brings 13.89% to 25.97% improvement under the MNAR setting. The superior performance of DAGAN is attributed to GAN-based architecture with both target mask generation and source data adaptation. Even with complicated missing patterns caused by MAR and MNAR, the architecture is effective and robust in achieving the data distribution preserving and missing pattern adaptive properties when adapting the source data. We also observe that, although some imputation methods also achieve good performance in some cases, e.g., Figure 6(b), DAGAN is more robust than those baseline approaches.

Figure 7 shows that data imputation approaches perform well under the MCAR setting. There are two reasons. First, the MCAR setting is simple and the distributional difference between source and target data is not severe. Second, most existing data imputation methods [36, 43, 50] consider MCAR as their basic missing setting and have proposed a bunch of optimization techniques for this setting. Note, however, that most of missing patterns in practice do not obey the MCAR assumption.



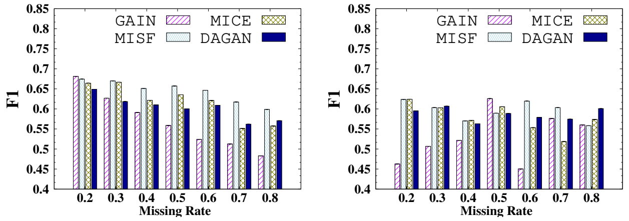
(a) EyeState dataset (b) Adult dataset

Figure 5: Robustness evaluation under NoOverlap-MNAR.



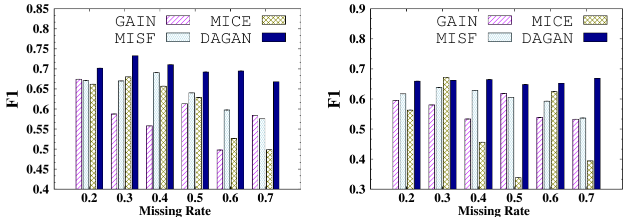
(a) EyeState dataset (b) Adult dataset

Figure 6: Robustness evaluation under NoOverlap-MAR.



(a) EyeState dataset (b) Adult dataset

Figure 7: Robustness evaluation under NoOverlap-MCAR.

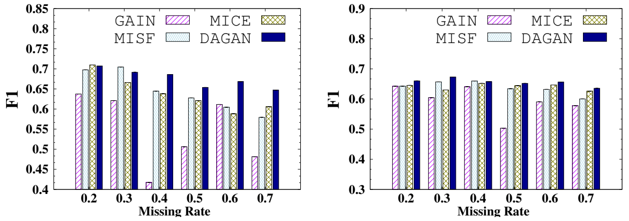


(a) EyeState dataset (b) Adult dataset

Figure 8: Robustness evaluation under Overlap-MNAR.

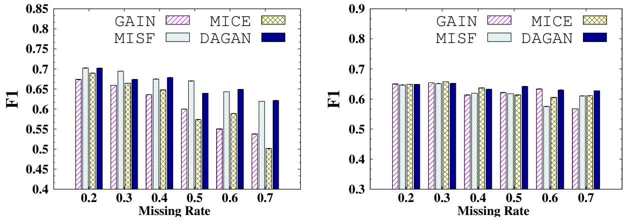
Note that we also conduct experiments under NoOverlap setting with low missing rates within the range between 0% and 10%. The result also validates that DAGAN is more robust than the baselines. Nevertheless, the improvement is less significant. The main reason is that deep learning models can tolerate dirty data to some extent by using strategies such as regularization [51], when missing rates are low. We put more discussion in our technical report [24].

Evaluation under Overlap noise injection setting. We also evaluate the approaches on the Overlap setting for injecting missing values. Unlike NoOverlap, we inject missing values to the *same* set of attributes in both source and target. However, to simulate noise shift, we fix the missing rate $p = 0.1$ in the source data, while vary the values of p from 0.2 to 0.7 in the target data. As observing



(a) `EyeState` dataset (b) `Adult` dataset

Figure 9: Robustness evaluation under `Overlap`-MAR.



(a) `EyeState` dataset (b) `Adult` dataset

Figure 10: Robustness evaluation under `Overlap`-MCAR.

Figures 8, 9 and 10, we find similar results to that of `NoOverlap` setting. The result validates that DAGAN is capable of handling noise shift for ML models in various missingness relationships.

Summary. To summarize, we compare the average performance of approaches across missing rates under different datasets and missing value injection setting, as shown Table 5, where notations "++", "+", "-" and "- -" represent the `1st`, `2nd`, `3rd` and `4th` methods in terms of F1 score respectively. We can see that DAGAN clearly outperforms existing imputation methods on MNAR and MAR (more complicated and more common in practice). Observed from the table, for MNAR and MAR, DAGAN is the winner in 7 out of 8 settings, while being the second best in only 1 setting. For MCAR, DAGAN achieves better performance than `GAIN` and `MICE` in most of the cases. It also achieves comparable results to `MISF`, i.e., the margin between `MISF` and DAGAN is 3.5%.

In practice, given a dataset with missing values, if it is "known" that the missing pattern is MCAR, we will suggest MISF, and if the missing pattern is MNAR or MAR, we will suggest DAGAN. Unfortunately, precisely deciding which missing value pattern (MCAR, MAR, or MNAR) a dataset has is a hard problem. Hence, a safe choice is to choose a method that is consistently good, i.e., the most robust. Table 5 clearly justifies the robustness of our approach.

6.6 Evaluation on Multi-Target Augmentation

This section evaluates the performance of DAGAN when conducting multi-target adaptive data augmentation. Specifically, we take $\mathcal{D}_\mathsf{s}$ as source data, and combine unlabeled datasets with different missing rates, i.e., $\mathcal{U}_\mathsf{t} = \bigcup_p \mathcal{U}_\mathsf{t}^p$ where $p \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$. Similarly, we also generate target test set by combining test sets with different missing rates, i.e., $\mathcal{D}_\mathsf{t} = \bigcup_p \mathcal{D}_\mathsf{t}^p$. We first evaluate model retraining strategies ERM and GDRO, by considering how the loss on the test dataset changes with training iterations. Figure 11 shows the experimental results on the `EyeState` dataset where legend like "test-0.2" indicates the loss on test data $\mathcal{D}_\mathsf{t}^p$ with missing rate $p = 0.2$. We have an interesting observation that GDRO performs much better than ERM to reduce the losses of different subgroups under all the missing settings. This is attributed to group-aware optimization scheme in GDRO. The optimization scheme focuses on optimizing the group with the maximum loss value, instead of the average loss. In such a way, it can enable losses of different groups to be simultaneously reduced. We also conduct experiments on the `Adult` dataset and find similar results. Please refer to our technical report [24].

Table 5: Comparison of Average Performance across Missing Rates under Various Missing Value Injection Settings, where notations "++", "+", "-" and "- -" represent the `1st`, `2nd`, `3rd` and `4th` methods in terms of F1 score respectively.

Missing	Dataset-Setting	GAIN	MISF	MICE	DAGAN
MNAR	`EyeState-Overlap`	-	+	- -	++
	`Adult-Overlap`	+	-	- -	++
	`EyeState-NoOverlap`	-	- -	+	++
	`Adult-NoOverlap`	-	- -	+	++
MAR	`EyeState-Overlap`	- -	+	-	++
	`Adult-Overlap`	- -	-	+	++
	`EyeState-NoOverlap`	-	- -	+	++
	`Adult-NoOverlap`	- -	-	++	+
MCAR	`EyeState-Overlap`	- -	++	-	+
	`Adult-Overlap`	-	- -	+	++
	`EyeState-NoOverlap`	- -	++	+	-
	`Adult-NoOverlap`	- -	++	-	+

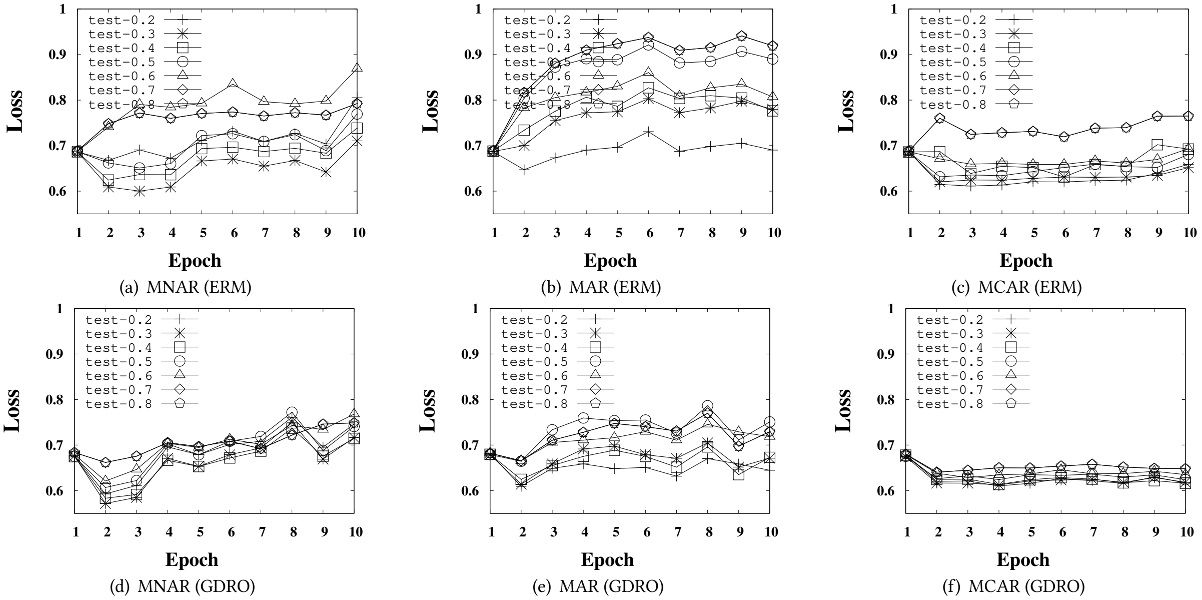
Table 6: Evaluating Model Retraining for Multi-ADA on F1.

Missing	Dataset	Multi-Clf	ERM	GDRO
MNAR	`EyeState`	**0.648**	0.645	0.646
	`Adult`	0.576	0.572	**0.632**
MAR	`EyeState`	**0.618**	0.566	0.616
	`Adult`	0.576	0.525	**0.605**
MCAR	`EyeState`	**0.603**	0.586	0.601
	`Adult`	**0.587**	0.506	0.579

Next, we compare DAGAN for serving *multiple* test sets with an alternative method discussed in Section 2. The alternative method trains multiple classifiers, where f_i^+ is used for test set $\mathcal{D}_\mathsf{t}^i$, which is different from our solution that only maintains one classifier f^+. For ease of presentation, we name the alternative method as Multi-Clf. The experimental result is shown in Table 6. We have the following observations. First, ERM degrades the performance compared with Multi-Clf, which validates our claim that the method of simply training classifier on the augmented data cannot achieve good performance. Second, GDRO achieves comparable F1 scores with and sometimes outperforms the Multi-Clf method. This result is encouraging because we only need to maintain one classifier in GDRO, which relieves user's burden on model deployment while still preserving prediction quality. The improvement brought by GDRO is due to the subgroup-aware optimization technique that considers to optimize the subgroup with the *largest loss* iteratively and thus GDRO can reduce the diversity among different subgroups.

7 RELATED WORK

Missing Data Imputation. We categorize existing methods into three groups: statistical-, discriminative model-, and generative model-based. Statistical methods include replacing the missing



(a) MNAR (ERM)

(b) MAR (ERM)

(c) MCAR (ERM)

(d) MNAR (GDRO)

(e) MAR (GDRO)

(f) MCAR (GDRO)

Figure 11: Evaluating model retraining strategies, ERM and GDRO, by varying training epochs on the EyeState dataset, where legend like "test-p" indicates the loss of our classifier on test data $\mathcal{D}_{\mathbf{t}}^{p}$ with missing rate p.

values with zero/mean/median value or mode value according to their meaning. These simple imputation methods are convenient to quickly validate claims but far from enough to be of high quality. Discriminative model-based methods include Multiple Imputation with Chained Equations (MICE) [36] and MissForest (MISF) [43]. They predict the missing values utilizing observed values through regression and random forest respectively. A recent work, Generative Adversarial Imputation Nets (GAIN) [50], is a generative model-based method based on GAN models. However, these traditional methods are mainly designed and proved effective for MCAR setting, so are less robust to more complex missing patterns such as MAR and MNAR. For supervised ML, the gap of different missing patterns between the source and the target is not considered by traditional approaches. Our work fills this gap.

Learned Data Cleaning. There have been learned approaches for error detection [30]. In our work, we consider values as explicitly missing and the detection is straightforward. There are also learned approaches for data repairing, e.g., GDR [48], SCAREd [47], and Baran [29]. GDR [48] and SCARED [47] learn data dependencies mainly for categorical values. Baran [29] ensembles the outputs from multiple error corrector models for repairing unnormalized values, dependency based errors, and outliers. HoloClean [38] is a statistical inference cleaning system, which leverages available quality rules, value correlations, reference data, and multiple other signals to build a probabilistic model for data repairing. The learned data repairing methods are not specially designed for repairing missing values, and they don't consider different missing patterns between the source and the target that is the main focus of this work, but they are complementary to our work.

Data Augmentation. Data augmentation is widely used in ML to obtain more training data [11, 13, 20, 21, 25, 27]. Traditional works apply heuristics, such as crops, rotations and flips to original images, for data augmentation [12, 13, 25, 37, 45]. As it is difficult to guarantee quality and diversity of the augmented data, some existing works consider using generative models to implement augmentation [37, 52]. However, few attention is paid to the adaptive relational data augmentation, especially for addressing the noise shift caused by missing data, as what we study in this paper.

8 CONCLUSION

We have presented a novel adaptive data augmentation framework for supervised ML. We have proposed a novel GAN-based neural networks, the DAGAN, to learn the noise mask from the target data and augment the source data by adapting them using the target mask. Empirically, we have shown that our method is more robust than the state-of-the-art data imputation methods.

Acknowledgment. This work was partly supported by National Key Research and Development Program of China (2020YFB2104101), NSF of China (62072461, 61632016, 61925205, U1911203), Huawei, and TAL Education.

REFERENCES

[1] 1996. Adult Data Set. https://archive.ics.uci.edu/ml/datasets/Adult.

[2] 2013. EyeState Data Set. http://archive.ics.uci.edu/ml/datasets/EEG+Eye+State.

[3] 2014. Ipums Data Set. https://www.openml.org/d/381.

[4] 2019. Okcupid Data Set. https://www.openml.org/d/41440.

[5] 2020. Welfare Data Set. https://www.cnrds.com/.

[6] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting Data Errors: Where are we and what needs to be done? *Proc. VLDB Endow.* 9, 12 (2016), 993–1004.

[7] Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *CoRR* abs/1701.07875 (2017).

[8] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Mach. Learn.* 79, 1-2 (2010), 151–175.

[9] Steffen Bickel, Michael Brückner, and Tobias Scheffer. 2009. Discriminative Learning Under Covariate Shift. *J. Mach. Learn. Res.* 10 (2009), 2137–2155.

[10] Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and V. S. Subrahmanian. 2019. FakeTables: Using GANs to Generate Functional Dependency Preserving Tables with Bounded Real Data. In *IJCAI*. 2074–2080.

[11] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2019. AutoAugment: Learning Augmentation Strategies From Data. In *CVPR*. 113–123.

[12] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin A. Riedmiller, and Thomas Brox. 2016. Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 38, 9 (2016), 1734–1747.

[13] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data Augmentation for Low-Resource Neural Machine Translation. In *ACL*, Regina Barzilay and Min-Yen Kan (Eds.). 567–573.

[14] Ju Fan, Tongyu Liu, Guoliang Li, Junyou Chen, Yuwei Shen, and Xiaoyong Du. 2020. Relational Data Synthesis using Generative Adversarial Networks: A Design Space Exploration. *Proc. VLDB Endow.* 13, 11 (2020), 1962–1975.

[15] Ju Fan, Tongyu Liu, Guoliang Li, Junyou Chen, Yuwei Shen, and Xiaoyong Du. 2020. Relational Data Synthesis using Generative Adversarial Networks: A Design Space Exploration. *CoRR* abs/2008.12763 (2020).

[16] Alireza Farhangfar, Lukasz A. Kurgan, and Jennifer G. Dy. 2008. Impact of imputation of missing values on classification error for discrete data. *Pattern Recognit.* 41, 12 (2008), 3692–3705.

[17] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. 2017. Domain-Adversarial Training of Neural Networks. In *Domain Adaptation in Computer Vision Applications*. 189–209.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*. 2672–2680.

[19] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Neural Information Processing Systems*, Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger (Eds.). 1106–1114.

[21] Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data Augmentation using Pre-trained Transformer Models. *CoRR* abs/2003.02245 (2020).

[22] Steven Cheng-Xian Li, Bo Jiang, and Benjamin M. Marlin. 2019. MisGAN: Learning from Incomplete Data with Generative Adversarial Networks. In *ICLR*. OpenReview.net.

[23] Zachary C. Lipton, Yu-Xiang Wang, and Alexander J. Smola. 2018. Detecting and Correcting for Label Shift with Black Box Predictors. In *ICML*, Vol. 80. 3128–3136.

[24] Tongyu Liu, Yinqing Luo, Ju Fan, Nan Tang, Guoliang Li, and Xiaoyong Du. 2020. Adaptive Data Augmentation for Supervised Learning over Missing Data. In *Technical Report*. https://github.com/ruclty/dagan/blob/master/dagan-tr.pdf.

[25] Xinghua Lu, Bin Zheng, Atulya Velivelli, and ChengXiang Zhai. 2006. Research Paper: Enhancing Text Categorization with Semantic-enriched Representation and Training Data Augmentation. *J. Am. Medical Informatics Assoc.* 13, 5 (2006), 526–535.

[26] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. 2018. Are GANs Created Equal? A Large-Scale Study. In *NeurIPS*. 698–707.

[27] Jizhou Luo, Wei Zhang, Shengfei Shi, Hong Gao, Jianzhong Li, Wei Wu, and Shouxu Jiang. 2019. FreshJoin: An Efficient and Adaptive Algorithm for Set Containment Join. *Data Sci. Eng.* 4, 4 (2019), 293–308.

[28] Lin Ma, Bailu Ding, Sudipto Das, and Adith Swaminathan. 2020. Active Learning for ML Enhanced Database Systems. In *SIGMOD*. 175–191.

[29] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective Error Correction via a Unified Context Representation and Transfer Learning. *Proc. VLDB Endow.* 13, 11 (2020), 1948–1961.

[30] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A Configuration-Free Error Detection System. In *SIGMOD*. 865–882.

[31] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. *CoRR* abs/1411.1784 (2014). arXiv:1411.1784 http://arxiv.org/abs/1411.1784

[32] Maryam M. Najafabadi, Flavio Villanustre, Taghi M. Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. 2015. Deep learning applications and challenges in big data analytics. *J. Big Data* 2 (2015), 1.

[33] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data Synthesis based on Generative Adversarial Networks. *PVLDB* 11, 10 (2018), 1071–1083.

[34] Therese Pigott. 2010. A Review of Methods for Missing Data. *Educational Research and Evaluation: An International Journal on Theory and Practice* 7 (08 2010), 353–383.

[35] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2018. Data Lifecycle Challenges in Production Machine Learning: A Survey. *SIGMOD Rec.* 47, 2 (2018), 17–28.

[36] Trivellore Raghunathan, James Lepkowski, John Hoewyk, and Peter Solenberger. 2000. A Multivariate Technique for Multiply Imputing Missing Values Using a Sequence of Regression Models. *Survey Methodology* 27 (11 2000).

[37] Alexander J. Ratner, Henry R. Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. 2017. Learning to Compose Domain-Specific Transformations for Data Augmentation. In *Neural Information Processing Systems*. 3236–3246.

[38] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proc. VLDB Endow.* 10, 11 (2017), 1190–1201.

[39] Stuart J. Russell and Peter Norvig. 2010. *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education.

[40] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. 2019. Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization. *CoRR* abs/1911.08731 (2019). arXiv:1911.08731 http://arxiv.org/abs/1911.08731

[41] Joseph Schafer and John Graham. 2002. Missing Data: Our View of the State of the Art. *Psychological Methods* 7 (06 2002), 147–177.

[42] Sebastian Schelter, Tammo Rukat, and Felix Bießmann. 2020. Learning to Validate the Predictions of Black Box Classifiers on Unseen Data. In *SIGMOD*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1289–1299. https://doi.org/10.1145/3318464.3380604

[43] Daniel J. Stekhoven and Peter Bühlmann. 2011. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics* 28, 1 (10 2011), 112–118. https://doi.org/10.1093/bioinformatics/btr597 arXiv:https://academic.oup.com/bioinformatics/article-pdf/28/1/112/583703/btr597.pdf

[44] Vladimir Vapnik. 1991. Principles of Risk Minimization for Learning Theory. In *NIPS*. 831–838.

[45] Jason W. Wei and Kai Zou. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *EMNLP-IJCNLP*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). 6381–6387.

[46] Richard Wu, Aoqian Zhang, Ihab F. Ilyas, and Theodoros Rekatsinas. 2020. Attention-based Learning for Missing Data Imputation in HoloClean. In *MLSys*, Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze (Eds.). mlsys.org. https://proceedings.mlsys.org/book/307.pdf

[47] Mohamed Yakout, Laure Berti-Équille, and Ahmed K. Elmagarmid. 2013. Don't be SCAREd: use SCalable Automatic REpairing with maximal likelihood and bounded changes. In *SIGMOD*. ACM, 553–564.

[48] Mohamed Yakout, Ahmed K. Elmagarmid, Jennifer Neville, Mourad Ouzzani, and Ihab F. Ilyas. 2011. Guided data repair. *Proc. VLDB Endow.* 4, 5 (2011), 279–289.

[49] Liu Yang, Steve Hanneke, and Jaime G. Carbonell. 2013. A theory of transfer learning with applications to active learning. *Mach. Learn.* 90, 2 (2013), 161–189.

[50] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GAIN: Missing Data Imputation using Generative Adversarial Nets. In *ICML (Proceedings of Machine Learning Research)*, Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 5675–5684.

[51] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *ICLR*.

[52] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *ICCV*. 2242–2251.

Algorithm 3: DAGAN Training

```
Input: $b$: batch size; $\alpha_g$: learning rate of $G_x$ and $G_m$; $\alpha_d$: learning
    rate of $D_x$ and $D_m$; $T$: number of training iterations
Output: $G_x$: data generator; $D_x$: data discriminator; $G_m$: mask
    generator; $D_m$: mask discriminator
Initialize parameters $\theta_{g_x}^{(0)}$,$\theta_{d_x}^{(0)}$ for $G_x$,$D_x$ and $\theta_{g_m}^{(0)}$,$\theta_{d_m}^{(0)}$ for $G_m$,$D_m$
for training iteration $t_1 = 1, 2, \ldots, T_g$ do
    /* Training discriminator $D_m$ */
    Sample $b$ noise samples $\{\boldsymbol{\sigma}^{(i)}\}_{i=1}^{b}$ from noise prior $p(\boldsymbol{\sigma})$
    Sample $b$ samples $\{\boldsymbol{x}_{\mathrm{t}}^{(i)}, \boldsymbol{m}_{\mathrm{t}}^{(i)}\}_{i=1}^{b}$ from target data $p(\boldsymbol{x}_{\mathrm{t}}, \boldsymbol{m}_{\mathrm{t}})$
    $\bar{\boldsymbol{m}} \leftarrow \{G_m(\boldsymbol{\sigma}^{(i)}, \boldsymbol{x}_{\mathrm{t}}^{(i)})\}_{i=1}^{b}$
    $\bar{g}_1 \leftarrow \nabla_{\theta_{d_m}} \frac{1}{b} \sum_{i=1}^{b} [D_m(\boldsymbol{m}^{(i)}, \boldsymbol{x}_{\mathrm{t}}^{(i)}) - D_m(\bar{\boldsymbol{m}}^{(i)}, \boldsymbol{x}_{\mathrm{t}}^{(i)})]$
    $\theta_{d_m}^{(t)} \leftarrow \theta_{d_m}^{(t-1)} + \alpha_d \cdot \mathsf{RMSProp}(\theta_{d_m}^{(t-1)}, \bar{g}_1)$
    /* Training discriminator $D_x$ */
    Sample $b$ noise samples $\{\boldsymbol{\sigma}^{(i)}\}_{i=1}^{b}$ from noise prior $p(\boldsymbol{\sigma})$
    Sample $b$ noise samples $\{\boldsymbol{z}^{(i)}\}_{i=1}^{b}$ from noise prior $p(\boldsymbol{z})$
    Sample $b$ samples $\{\boldsymbol{x}_{\mathrm{t}}^{(i)}, \boldsymbol{m}_{\mathrm{t}}^{(i)}\}_{i=1}^{b}$ from target data $p(\boldsymbol{x}_{\mathrm{t}}, \boldsymbol{m}_{\mathrm{t}})$
    Sample $b$ samples $\{\boldsymbol{x}_{\mathrm{s}}^{(i)}, \boldsymbol{m}_{\mathrm{s}}^{(i)}\}_{i=1}^{b}$ from source data $p(\boldsymbol{x}_{\mathrm{s}}, \boldsymbol{m}_{\mathrm{s}})$
    $\tilde{\boldsymbol{x}} \leftarrow \{G_x(\boldsymbol{z}^{(i)}, \boldsymbol{x}_{\mathrm{s}}^{(i)}, \boldsymbol{m}_{\mathrm{s}}^{(i)})\}_{i=1}^{b}$
    $\tilde{\boldsymbol{m}}_{\mathrm{t}} \leftarrow \{G_m(\boldsymbol{\sigma}^{(i)}, \tilde{\boldsymbol{x}}^{(i)})\}_{i=1}^{b}$
    $\tilde{\boldsymbol{x}}_{\mathrm{t}} \leftarrow \{\psi(\tilde{\boldsymbol{x}}^{(i)}, \tilde{\boldsymbol{m}}_{\mathrm{t}}^{(i)})\}_{i=1}^{b}$
    $\bar{g}_2 \leftarrow \nabla_{\theta_{d_x}} \frac{1}{b} \sum_{i=1}^{b} [D_x(\boldsymbol{x}_{\mathrm{t}}^{(i)}) - D_x(\tilde{\boldsymbol{x}}_{\mathrm{t}}^{(i)})]$
    $\theta_{d_x}^{(t)} \leftarrow \theta_{d_x}^{(t-1)} + \alpha_d \cdot \mathsf{RMSProp}(\theta_{d_x}^{(t-1)}, \bar{g}_2)$
    /* Training generators $G_m$ and $G_x$ */
    Sample $b$ noise samples $\{\boldsymbol{\sigma}^{(i)}\}_{i=1}^{b}$ from noise prior $p(\boldsymbol{\sigma})$
    Sample $b$ noise samples $\{\boldsymbol{z}^{(i)}\}_{i=1}^{b}$ from noise prior $p(\boldsymbol{z})$
    Sample $b$ samples $\{\boldsymbol{x}_{\mathrm{t}}^{(i)}, \boldsymbol{m}_{\mathrm{t}}^{(i)}\}_{i=1}^{b}$ from target data $p(\boldsymbol{x}_{\mathrm{t}}, \boldsymbol{m}_{\mathrm{t}})$
    Sample $b$ samples $\{\boldsymbol{x}_{\mathrm{s}}^{(i)}, \boldsymbol{m}_{\mathrm{s}}^{(i)}\}_{i=1}^{b}$ from source data $p(\boldsymbol{x}_{\mathrm{s}}, \boldsymbol{m}_{\mathrm{s}})$
    $\tilde{\boldsymbol{x}} \leftarrow \{G_x(\boldsymbol{z}^{(i)}, \boldsymbol{x}_{\mathrm{s}}^{(i)}, \boldsymbol{m}_{\mathrm{s}}^{(i)})\}_{i=1}^{b}$
    $\tilde{\boldsymbol{m}}_{\mathrm{t}} \leftarrow \{G_m(\boldsymbol{\sigma}^{(i)}, \tilde{\boldsymbol{x}}^{(i)})\}_{i=1}^{b}$
    $\tilde{\boldsymbol{x}}_{\mathrm{t}} \leftarrow \{\psi(\tilde{x_s}^{(i)}, \tilde{\boldsymbol{m}}_{\mathrm{t}}^{(i)})\}_{i=1}^{b}$
    $\bar{\boldsymbol{m}} \leftarrow \{G_m(\boldsymbol{\sigma}^{(i)}, \boldsymbol{x}_{\mathrm{t}}^{(i)})\}_{i=1}^{b}$
    $\bar{g}_3 \leftarrow -\nabla_{\theta_{g_x}} \frac{1}{b} \sum_{i=1}^{b} D_x(\tilde{\boldsymbol{x}}_{\mathrm{t}}^{(i)})$
    $\theta_{g_x}^{(t)} \leftarrow \theta_{g_x}^{(t-1)} - \alpha_g \cdot \mathsf{RMSProp}(\theta_{g_x}^{(t-1)}, \bar{g}_3)$
    $\bar{g}_4 \leftarrow -\nabla_{\theta_{g_m}} \frac{1}{b} \sum_{i=1}^{b} [D_x(\tilde{\boldsymbol{x}}_{\mathrm{t}}^{(i)}) + D_m(\bar{\boldsymbol{m}}^{(i)}, \boldsymbol{x}_{\mathrm{t}}^{(i)})]$
    $\theta_{g_m}^{(t)} \leftarrow \theta_{g_m}^{(t-1)} - \alpha_g \cdot \mathsf{RMSProp}(\theta_{g_m}^{(t-1)}, \bar{g}_4)$
return $G_x, D_x, G_m, D_m$
```

A APPENDIX

A.1 Algorithm of DAGAN Training

Algorithm 3 provides the detailed pseudo-code of the training process of DAGAN (a simple version of the code can be found in Algorithm 1).

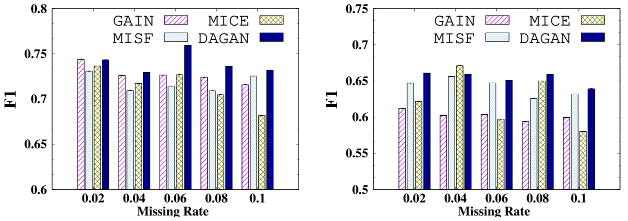
A.2 Additional Experiments

A.2.1 Details of Real-World Datasets. We utilize three real-world datasets with *real*, instead of injected, missing values. As described in Section 6.1, we partition the datasets according to timestamp to obtain the source and the target data. We present detailed information of the attributes with missing values on these datasets.

(1) Ipums Dataset: There are 17 missing attributes in source data. We list the attributes together with their missing rates as follows: ownershg (1.8%), chborn (59.1%), edurec (4.3%), schltype (71.0%), empstatg (24.2%), labforce (24.2%), occ1950 (42.5%), ind1950 (42.6%), classwkg (42.5%), wkswork2 (49.3%), hrswork2 (55.5%), yrlastwk (68.7%), workedyr (24.2%), migrat5g (52.8%), migplac5 (52.8%), vetstat (24.3%), tranwork (55.5%). On the other hand, there are 18 missing attributes in target data: ownershg (1.9%), chborn (61.2%), school (4.8%), edurec (4.8%), schltype (4.8%), empstatg (23.9%), labforce (23.9%), occ1950 (41.5%), ind1950 (41.5%), classwkg (41.5%), wkswork2 (47.5%), hrswork2 (54.1%), yrlastwk (69.8%), workedyr (23.9%), migrat5g (8.0%), migplac5 (51.4%), vetstat (23.9%), tranwork (54.1%).

(2) Okcupid Dataset: There are 13 missing attributes in source data: body_type (9.4%), diet (46.8%), drinks (3.2%), drugs (20.7%), education (7.8%), ethnicity (8.1%), income (74.8%), offspring (67.7%), pets (30.6%), religion (30.3%), sign (15.2%), smokes (7.0%), speaks (0.1%). There are 13 missing attributes in target data: body_type (6.6%), diet (33.0%), drinks (2.9%), drugs (24.2%), education (6.3%), ethnicity (7.7%), income (80.1%), offspring (50.5%), pets (28.7%), religion (29.5%), sign (15.1%), smokes (7.1%), speaks (0.04%).

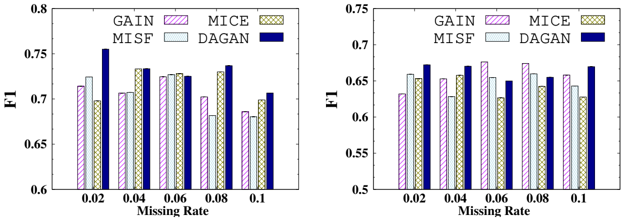
(3) Welfare Dataset: There are 2 missing attributes in source data: Lyfb (34.6%), Aoexp (0.1%). There are 5 missing attributes in target data: Lyfb (15.1%), Atotalexp (0.2%), Pwelexp (0.1%), Wwexp (0.4%), Aoexp (0.3%).



(a) EyeState dataset

(b) Adult dataset

Figure 12: Robustness evaluation under NoOverlap-MNAR setting with low missing rates.

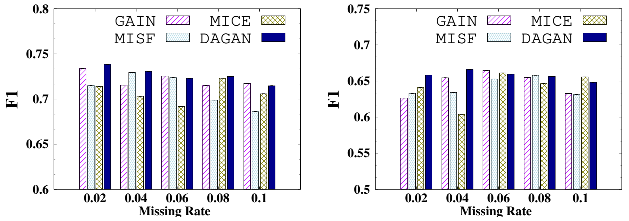


(a) EyeState dataset

(b) Adult dataset

Figure 13: Robustness evaluation under NoOverlap-MAR setting with low missing rates.

A.2.2 Evaluation on Robustness Evaluation. We also evaluate the robustness of DAGAN by comparing with the existing data imputation methods under NoOverlap setting with low missing rates. Specifically, we inject missing values with missing rate p to the first half of attributes in the source and the last half of attributes in the target respectively. We vary the values of missing rate p within the range between 0% and 10%. Figures 12, 13 and 14 show



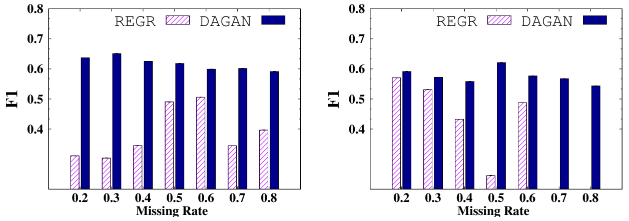
(a) EyeState dataset

(b) Adult dataset

Figure 14: Robustness evaluation under NoOverlap-MCAR setting with low missing rates.

Table 7: Comparison of Average Performance under Different Missing Value Injection Settings with Low Missing Rates where notations "++", "+", "-" and "- -" represent the 1st, 2nd, 3rd and 4th methods in terms of F1 score respectively.

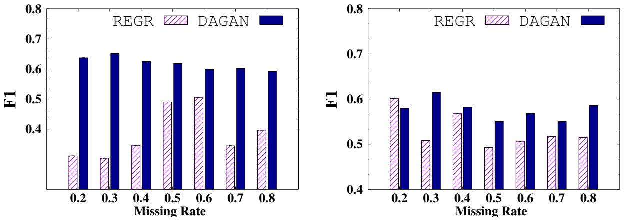
Missing	Dataset	GAIN	MISF	MICE	DAGAN
MNAR	EyeState	+	-	- -	++
	Adult	- -	+	-	++
MAR	EyeState	-	- -	+	++
	Adult	+	-	- -	++
MCAR	EyeState	+	-	- -	++
	Adult	+	-	- -	++



(a) EyeState dataset

(b) Adult dataset

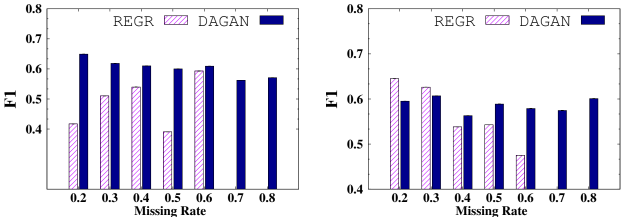
Figure 15: Comparing DAGAN with regression imputation approach under difficult missing scenario MNAR.



(a) EyeState dataset

(b) Adult dataset

Figure 16: Comparing DAGAN with regression imputation approach under difficult missing scenario MAR.



(a) EyeState dataset

(b) Adult dataset

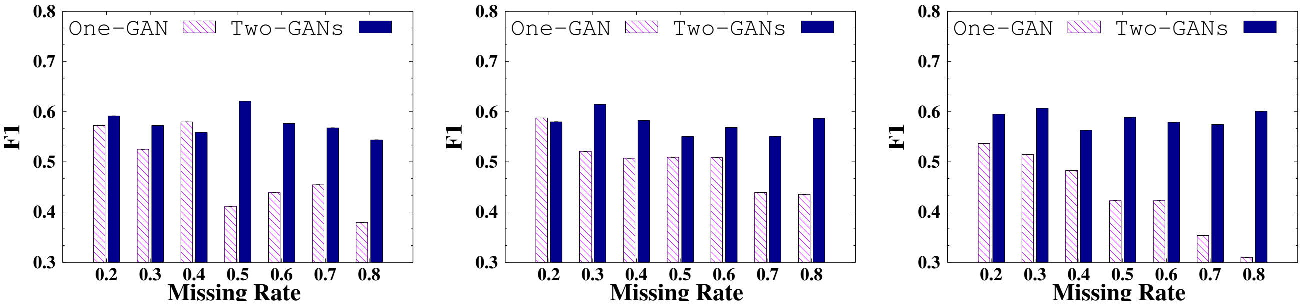
Figure 17: Comparing DAGAN with regression imputation approach under simple missing scenario MCAR.

the experimental results, which also validates that DAGAN is more robust than the baselines. A clear way to understand the comparison of average performance of different approaches under different missing patterns is summarized in Table 7, where we use "++", "+", "- -" and "-" to denote the best, second best, the worst and second worst results, respectively. The experimental result also validates that DAGAN outperforms the baselines. Nevertheless, we also find that the improvement is less significant, with the main reason that deep learning models are robust to tolerate dirty data by using strategies such as regularization [51], when missing rates are low.

A.2.3 Comparison with More Baselines. To further validate the superiority of adaptive data augmentation, we compare DAGAN with the following alternative approach REGR. In REGR, we first use the *fully cleaned* subset of source data to learn a completion (i.e., missing data imputation) model. Then, we use the model to impute missing values in both the source data and the target data. Next, we train our ML model on the source and evaluate the model on the target test data. In particular, we use a regression model to implement the completion model. Figures 15, 16 and 17 show the experimental results under MNAR, MAR and MCAR. We can see that DAGAN clearly outperforms the baseline. This is mainly attributed to the noise shift between source and target, where a completion model trained on fully cleaned source might not be able to effectively impute the target. In contrast, our DAGAN approach adaptively augments source data by the missing pattern of the target, and thus alleviate the effect of dataset shift.

We empirically compare our approach with an alternative of ONE GAN that takes a source tuple as input and outputs a fake target tuple. As shown in Figure 18, the experimental results show that the baseline achieves inferior performance. The main reason is that one GAN might be ineffective to recover mixture of both data and noise distributions.

A.2.4 Evaluation on Model Retraining. We evaluate model retraining strategies ERM and GDRO, by considering how the loss on the test dataset changes with training iterations on the Adult dataset. Note that the results on the EyeState dataset can be found in Section 6.6 Figure 19 validates that GDRO performs much better than ERM to reduce the losses of different subgroups under all the missing settings on the Adult dataset. This is attributed to group-aware optimization scheme in GDRO. The optimization scheme focuses on optimizing the group with the maximum loss value, instead of the average loss. In such a way, it can enable losses of different groups to be simultaneously reduced.

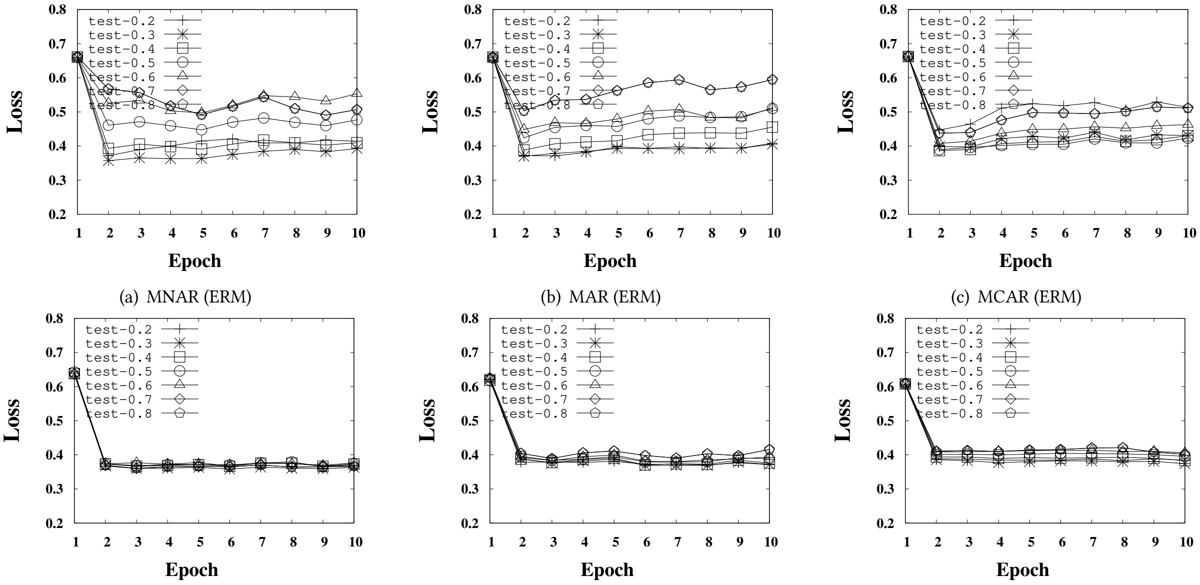


(a) MNAR Setting

(b) MAR Setting

(c) MCAR Setting

Figure 18: Comparing with an alternative model design with one GAN.



(a) MNAR (ERM)

(b) MAR (ERM)

(c) MCAR (ERM)

(d) MNAR (GDRO)

(e) MAR (GDRO)

(f) MCAR (GDRO)

Figure 19: Evaluating model retraining strategies, ERM and GDRO, by varying training epochs on the `Adult` dataset, where legend like "test-p" indicates the loss of our classifier on test data $\mathcal{D}_{\mathbf{t}}^{p}$ with missing rate p.