Efficient and Effective Data Imputation with Influence Functions

Xiaoye Miao*[1], Yangyang Wu*[2], Lu Chen†[3], Yunjun Gao†[4], Jun Wang‡[5], Jianwei Yin*†[6]
*Center for Data Science, Zhejiang University, Hangzhou, China
†College of Computer Science, Zhejiang University, Hangzhou, China
‡Information Hub, The Hong Kong University of Science and Technology, Hong Kong, China
{miaoxy[1], zjuwuyy[2], luchen[3], gaoyj[4], zjuyjw[6]}@zju.edu.cn jwangfx[5]@connect.ust.hk

ABSTRACT

Data imputation has been extensively explored to solve the missing data problem. The dramatically rising volume of missing data makes the training of imputation models computationally infeasible in real-life scenarios. In this paper, we propose an efficient and effective data imputation system with *influence functions*, named EDIT, which quickly trains a parametric imputation model with representative samples under imputation accuracy guarantees. EDIT mainly consists of two modules, i.e., an *imputation influence evaluation* (IIE) module and a *representative sample selection* (RSS) module. IIE leverages the influence functions to estimate the effect of (in)complete samples on the prediction result of parametric imputation models. RSS builds a minimum set of the high-effect samples to satisfy a user-specified imputation accuracy. Moreover, we introduce a weighted loss function that drives the parametric imputation model to pay more attention on the high-effect samples. Extensive experiments upon ten state-of-the-art imputation methods demonstrate that, EDIT adopts only about 5% samples to speed up the model training by 4x in average with more than 11% accuracy gain.

PVLDB Reference Format:
Xiaoye Miao, Yangyang Wu, Lu Chen, Yunjun Gao, Jun Wang, Jianwei Yin. Efficient and Effective Data Imputation with Influence Functions. PVLDB, 15(3): 624 - 632, 2022.
doi:10.14778/3494124.3494143

1 INTRODUCTION

Missing data is a pervasive problem in many applications [3, 26, 28, 33, 34, 37, 41], such as, medical logs, meteorology records, advertising data, etc. Data may be missing due to many reasons like improper collection [40], lost records [3], or privacy concerns [31]. As a consequence, the missing data problem causes serious challenges for effective data analysis [21].

To handle the data missing problem, a substantial amount of research [12, 24, 30, 45] on the data imputation has been carried out, with the goal of imputing missing values using correct ones. Earlier simple statistical imputation methods [1, 12, 18, 38] directly substitute missing values with statistics or the most similar ones among the data. In contrast, the parametric imputation algorithms [27, 36, 45, 46], using the machine learning or deep learning models with the gradient descent style algorithms [32] to learn the observed data distribution, get better performance.

As data collection becomes the norm, the volume of missing data dramatically increases. For example, a real-world COVID-19 community mobility dataset *Mobility* [39] contains 2,268,105 incomplete samples with more than 5 million missing values. Although existing parametric imputation methods achieve better imputation performance than that of statistical ones, they consume extremely high training cost over the massive missing data, especially on calculating gradient over the entire dataset. Our experimental results show that, almost all parametric imputation methods take more than 10^5 seconds on training over the *million-size* incomplete data. Moreover, some training samples (e.g., samples contain errors) may have *little or even negative* effects on the parametric imputation model prediction, incurring redundant training [20, 42]. In fact, in many real-life scenarios, such as e-commerce, transportation science, and health-care, the efficient and effective data imputation over the massive missing data is necessary and indispensable.

As a result, in order to apply the parametric imputation methods to specified real-life scenarios, two major challenges have to be tackled: (i) How to evaluate the effect of incomplete samples on the prediction of parametric imputation models. (ii) Is it possible to build a minimum training set containing the high-effect samples to speed up the imputation model training under accuracy guarantees.

Therefore, in this paper, we propose an efficient and effective data imputation system EDIT. It evaluates the effect of incomplete samples on parametric imputation models by using the influence functions. It seeks the highest-effect incomplete samples for efficient and accuracy-guaranteed imputation model training. EDIT consists of the *imputation influence evaluation* (IIE) and *representative sample selection* (RSS) modules. In terms of the first challenge, the IIE module of EDIT introduces a novel concept of *influence power* that leverages the influence functions to evaluate the effect of (in)complete samples on the imputation model's prediction. Regarding the second challenge, EDIT designs the RSS module to construct a *minimum representative set* with the highest influence power samples, to enable the trained model to satisfy a user-specified imputation accuracy. Thus, EDIT employs a minimum representative sample set to make parametric imputation models *efficient* and *scalable* on massive missing data. EDIT also presents a weighted imputation loss function that utilizes the influence power of (in)complete samples to achieve higher imputation accuracy. In summary, our contributions of this paper are described as follows.

- We propose an efficient and effective data imputation system EDIT for fast parametric imputation model training under accuracy guarantees. To the best of our knowledge, this is the first proposal to accelerate imputation models.

Proceedings of the VLDB Endowment, Vol. 15, No. 3 ISSN 2150-8097.
doi:10.14778/3494124.3494143

- The IIE module of EDIT employs the influence functions to capture the influence power of the (in)complete sample on the prediction result of imputation model when adding (resp. deleting) samples into (resp. from) training data.
- The RSS module of EDIT is responsible for constructing a *minimum representative set*, containing the high-influence samples, to meet a user-specified imputation accuracy. It thus significantly saves the model training cost. We develop an effective imputation loss function for EDIT, which drives the imputation model to focus more on the samples with high influence powers.
- Extensive experiments over several datasets demonstrate the performance enhancement of EDIT, compared with the ten state-of-the-art parametric imputation methods.

The rest of the paper is organized as follows. We introduce the background in Section 2. Section 3 gives an overview of the proposed system EDIT. We elaborate the IIE and RSS modules in Section 4 and Section 5, respectively. Section 6 reports the experimental results and findings. Finally, we conclude this work in Section 7.

2 BACKGROUND

2.1 Existing Imputation Methods

Existing imputation algorithms can be categorized into two groups: statistical ones and parametric ones. The statistical imputation methods substitute missing values with the statistics [12], or the most similar ones among training data [1, 38]. They ignore the data distribution analysis, and thus have a limited ability to impute data.

In contrast, the parametric imputation solution is to train a parametric model [44, 47] (i.e., machine learning one or deep learning one) to learn the data distribution from the observed data. This solution employs the gradient descent techniques to estimate missing values. It gets better imputation performance than the statistical ones. On the one hand, the machine learning imputation approaches calculate the model gradient over the entire dataset to estimate missing values, including decision tree models XGBoost imputation [7], MissFI (MissForest imputation) [36], and Baran [23], and regression models MICE (multivariate imputation by chained equations) [30] and imputation via individual model [46]. However, the incomplete dataset may be too large to fit in memory among these methods.

On the other hand, some deep learning models [11, 22] are leveraged to solve the data imputation problem, such as multi-layer perceptron (MLP) [48], autoencoder (AE) [16], and generative adversarial network (GAN) [2, 15]. In particular, the *MLP-based* imputation ones include DataWig [4] and RRSI (round-robin Sinkhorn imputation) [5]. The *AE-based* imputation ones contains MIDAE (multiple imputation denoising AE) [14], VAEI (variational AE imputation) [25], HIVAE (heterogeneous incomplete VAE) [27], and MIWAE (missing data importance-weighted AE) [6]. The *GAN-based* imputation ones are composed of GINN (graph imputation neural network) [35] and GAIN (generative adversarial imputation network) [45]. The above methods can train the imputation models over the massive data, where the gradient is calculated over a series of small random partitions of the dataset. However, both the iteration times and training cost are dramatically increasing with the rising volume of missing data.

Table 1: Symbols and description

Symbol	Description
$\mathbf{X}$	an input incomplete dataset (stored in a matrix)
$\mathbf{X}^N$ and $\mathbf{H}$	the training set and holdout set (stored in matrices)
$\mathbf{X}^0$ and $\mathbf{X}^c$	the initial set and candidate set (stored in matrices)
$\mathbf{x}_i$	the i-th sample in the incomplete dataset $\mathbf{X}$
$\mathbf{M}$	the mask matrix w.r.t. $\mathbf{X}$
$\mathcal{M}$	an imputation model
$\bar{\mathbf{X}}$ and $\hat{\mathbf{X}}$	the reconstructed matrix and imputed matrix of $\mathbf{X}$
$\mathcal{I}_{\mathbf{H}}(\mathbf{x}_i)$	the influence power of the sample $\mathbf{x}_i$

In summary, the parametric imputation algorithms outperform the statistical ones. However, due to the high training cost, the parametric imputation methods are faced with a big challenge to deal with massive missing data imputation. Therefore, our proposed EDIT system, with the ultimate goal of making imputation models practical, aims to speed up the training of parametric imputation models with representative samples under accuracy guarantees.

2.2 Problem Definition

The input incomplete dataset is stored in a matrix $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_s)^\top$, in which each data sample $\mathbf{x}_i$ is in the form $(x_{i1}, \cdots, x_{id})$ within a d-dimensional space. For encoding its missing information, we define a mask matrix $\mathbf{M} = \{\mathbf{m}_1, \cdots, \mathbf{m}_s\}^\top$, where each mask vector $\mathbf{m}_i = (m_{i1}, \cdots, m_{id})$ corresponds to a data sample $\mathbf{x}_i$. In particular, m_{ij} takes value in $\{0, 1\}$, $i = 1, \cdots, s$, and $j = 1, \cdots, d$; $m_{ij} = 1$ (resp. 0) *iff* the j-th dimension is observed (resp. missing). Table 1 lists the frequently used symbols throughout this paper.

We formalize the *missing data imputation* problem over the incomplete dataset $\mathbf{X}$ with the mask matrix $\mathbf{M}$ in Definition 1.

Definition 1. **(Missing data imputation)**. *Given an incomplete dataset* $\mathbf{X}$ *with its mask matrix* $\mathbf{M}$*, the missing data imputation problem aims to build an imputation model* $\mathcal{M}$ *to find appropriate values for missing components in* $\mathbf{X}$*, i.e., the* imputed *matrix*

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

where $\odot$ *is the element-wise multiplication and* $\bar{\mathbf{X}} = \mathcal{M}(\mathbf{X})$ *is the reconstructed matrix predicted by* $\mathcal{M}$ *over* $\mathbf{X}$*. In this way, the imputation model* $\mathcal{M}$*, (i) makes* $\hat{\mathbf{X}}$ *as* close *to the real* complete *dataset (if it exists) as possible, or (ii) helps downstream prediction tasks to achieve better performance if adopting* $\hat{\mathbf{X}}$ *than that only with original one* $\mathbf{X}$*.*

In general, for a parametric imputation model $\mathcal{M}$ with parameters $\theta \in \Theta$, the imputation loss function can be defined as

$$\mathcal{L}(\mathbf{X}, \mathbf{M}, \theta) = \frac{1}{s} \sum_{i=1}^{s} \frac{||\mathbf{m}_i \odot (\bar{\mathbf{x}}_i - \mathbf{x}_i)||_2^2}{2||\mathbf{m}_i||_2^2} \tag{2}$$

where $||\cdot||_2$ is the two-norm function and $\bar{\mathbf{x}}$ in $\bar{\mathbf{X}}$ is predicted by $\mathcal{M}$. Thus, the loss minimizer is given by $\hat{\theta} \stackrel{\text{def}}{=} \arg\min_{\theta \in \Theta} \mathcal{L}(\mathbf{X}, \mathbf{M}, \theta)$. As a result, the parametric model employs the model $\mathcal{M}$ with optimized $\hat{\theta}$ to impute missing values in $\mathbf{X}$ via using Eq. 1.

Finally, our study mission in this paper is to empower the parametric imputation model $\mathcal{M}$ in efficiency and effectiveness for the massive missing data, such that for the optimized model $\mathcal{M}^*$, (i) the training cost is minimized, and (ii) the imputation accuracy is at least as good as that derived from the original model $\mathcal{M}$.

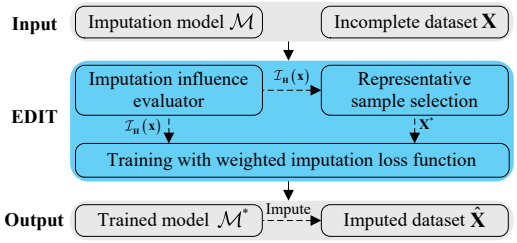


Figure 1: The architecture of EDIT

3 SYSTEM OVERVIEW

In this section, we present an overview of the proposed EDIT system. It is mainly composed of an *imputation influence evaluation* (IIE) module and a *representative sample selection* (RSS) module.

Figure 1 illustrates the workflow of the EDIT system. The inputs of EDIT include a parametric imputation model $\mathcal{M}$ and an incomplete dataset $\mathbf{X}$. It outputs the optimized model $\mathcal{M}^*$ and the data $\hat{\mathbf{X}}$ imputed by $\mathcal{M}^*$. Specifically, EDIT first utilizes the IIE module to estimate the *influence power*, denoted by $\mathcal{I}_{\mathbf{H}}(\mathbf{x})$, of the (in)complete sample $\mathbf{x}$ over the holdout set $\mathbf{H}$. Then, EDIT employs the RSS module to build a *minimum representative set* $\mathbf{X}^*$ containing the samples with the highest influence powers for satisfying an *imputation accuracy*. Finally, EDIT trains the imputation model with a newly introduced loss function weighted by the influence power.

The pseudo-code for the EDIT system is given in Algorithm 1. EDIT first randomly partitions the input dataset $\mathbf{X} \in \mathbb{R}^{s\times d}$ (with its mask matrix $\mathbf{M}$ encoding the missing information in $\mathbf{X}$) into an initial set $\mathbf{X}^0 \in \mathbb{R}^{n_0\times d}$ (with the initial mask matrix $\mathbf{M}^0$), a candidate dataset $\mathbf{X}^c \in \mathbb{R}^{(N-n_0)\times d}$ (with the candidate mask matrix $\mathbf{M}^c$), and a holdout dataset $\mathbf{H} \in \mathbb{R}^{(s-N)\times d}$ (with the holdout mask matrix $\mathbf{M}^h$), where the training dataset $\mathbf{X}^N = \mathbf{X}^0 \cup \mathbf{X}^c$ (with the training mask matrix $\mathbf{M}^N = \mathbf{M}^0 \cup \mathbf{M}^c$). EDIT invokes the parametric imputation model $\mathcal{M}$ to train an initial model $\mathcal{M}^0$ using $\mathbf{X}^0$ and $\mathbf{M}^0$. Then, the IIE module uses the influence functions to evaluate the influence power of each (in)complete sample $\mathbf{x} \in \mathbf{X}^N$ on the imputation prediction of $\mathcal{M}^0$ for $\mathbf{H}$ (when adding/deleting $\mathbf{x}$ into/from $\mathbf{X}^0$). The details will be elaborated in Section 4 (lines 1-3).

Next, EDIT consults the RSS module to estimate the imputation loss $\hat{\mathcal{L}}_{\mathbf{H}}(\mathbf{X}^c)$ of $\mathcal{M}^0$ over $\mathbf{H}$, when $\mathbf{X}^N$ is used for training. The user-specified imputation loss $\hat{\mathcal{L}}^{\alpha}_{\mathbf{H}}(\mathbf{X}^c)$ is calculated by $\frac{1}{\alpha}\cdot\hat{\mathcal{L}}_{\mathbf{H}}(\mathbf{X}^c)$. EDIT chooses the samples from $\mathbf{X}^c$ with the highest positive influence powers to form a minimum representative sample set $\mathbf{X}^*$ (with the mask matrix $\mathbf{M}^*$) to satisfy the user-specified imputation loss. The stop criterion of the sample selection is that, the newly estimated imputation loss (via adding all chosen samples into the training set) is exactly not worse than the user-specified loss $\hat{\mathcal{L}}^{\alpha}_{\mathbf{H}}(\mathbf{X}^c)$. The RSS module will be detailed in Section 5 (lines 4-5). Thereafter, EDIT trains $\mathcal{M}^0$ with a weighted imputation loss function using the data $\mathbf{X}^f (= \mathbf{X}^0 \cup \mathbf{X}^*)$ and mask matrix $\mathbf{M}^f (= \mathbf{M}^0 \cup \mathbf{M}^*)$. Eventually, EDIT returns the optimized model $\mathcal{M}^*$ and imputed data $\hat{\mathbf{X}}$ (lines 6-8).

4 IIE MODULE

In this section, we present how to estimate the *influence power* of sample(s), i.e., the change in the imputation loss of the initial model.

Algorithm 1: The EDIT System

Input: an incomplete dataset $\mathbf{X}$ with its mask matrix $\mathbf{M}$, an initial sample size n_0, a training sample size N, a hype-parameter α, and a parametric imputation model $\mathcal{M}$

Output: the trained imputation model $\mathcal{M}^*$ and imputed dataset $\hat{\mathbf{X}}$

1: divide $\mathbf{X}$ (with $\mathbf{M}$) into an initial dataset $\mathbf{X}^0$ (with $\mathbf{M}^0$), a candidate dataset $\mathbf{X}^c$ (with $\mathbf{M}^c$), and a holdout dataset $\mathbf{H}$ (with $\mathbf{M}^h$), i.e., $\mathbf{X} = \mathbf{X}^0 \cup \mathbf{X}^c \cup \mathbf{X}^h$ and $\mathbf{M} = \mathbf{M}^0 \cup \mathbf{M}^c \cup \mathbf{M}^h$

2: train the initial model $\mathcal{M}^0$ with $\mathbf{X}^0$ and $\mathbf{M}^0$ to obtain parameters $\hat{\theta}_0$

/* **Imputation influence evaluation** */

3: calculate the influence power $\mathcal{I}_{\mathbf{H}}(\mathbf{x})$ for each $\mathbf{x} \in \mathbf{X}^N (= \mathbf{X}^0 \cup \mathbf{X}^c)$

/* **Representative sample selection** */

4: derive the user-specified imputation loss $\hat{\mathcal{L}}^{\alpha}_{\mathbf{H}}(\mathbf{X}^c)$ over $\mathbf{H}$

5: build a minimum representative sample set $\mathbf{X}^*$ (with $\mathbf{M}^*$) via selecting the samples from $\mathbf{X}^c$ to satisfy $\hat{\mathcal{L}}^{\alpha}_{\mathbf{H}}(\mathbf{X}^c)$

/* **Training with weighted imputation loss function** */

6: train $\mathcal{M}_0$ with a weighted imputation loss function over $\mathbf{X}^f = \mathbf{X}^0 \cup \mathbf{X}^*$ and $\mathbf{M}^f = \mathbf{M}^0 \cup \mathbf{M}^*$ to obtain the optimized $\mathcal{M}^*$

7: estimate missing values by $\mathcal{M}^*$ to obtain the imputed matrix $\hat{\mathbf{X}}$

8: **return** $\mathcal{M}^*$ and $\hat{\mathbf{X}}$

Let us begin by employing the *influence function*, a classic technique from the robust statistics [20, 42], to evaluate the perturbation in the parameters of the initial model $\mathcal{M}^0$ when adding/deleting one sample $\mathbf{x} \in \mathbf{X}^N$ into/from the initial set $\mathbf{X}^0$. Specifically, inspired by the studies [20, 42], we formally define the influence function for the parametric imputation models in Definition 2. It evaluates the parameter perturbation, i.e., $\hat{\theta}_0 - \hat{\theta}_{\pm\mathbf{x}}$, with $\hat{\theta}_{\pm\mathbf{x}} = \arg\min_{\theta\in\Theta} \mathcal{L}(\{\mathbf{X}^0 \pm \{\mathbf{x}\}\}, \{\mathbf{M}^0 \pm \{\mathbf{m}\}\}, \theta)$, when upweighting the observed values in $\mathbf{x}$ by an infinitesimal amount. This estimation method on parameter perturbation avoids retraining $\mathcal{M}^0$ with parameters $\hat{\theta}_0$ (that is prohibitively expensive and unrealistic, especially when considering the massive missing data).

Definition 2. **(Influence function)**. *Given a sample $\mathbf{x}$ with its mask vector $\mathbf{m}$, the influence function $\mathcal{I}_\theta(\mathbf{x})$ is to compute the effect of upweighting the observed values in $\mathbf{x}$ with some small ϵ (on the parameters of an imputation model $\mathcal{M}$), getting parameters $\hat{\theta}_{\epsilon,\mathbf{x}} \stackrel{def}{=} \arg\min_{\theta\in\Theta} \mathcal{L}(\mathbf{X}^0, \mathbf{M}^0, \theta) + \epsilon\mathcal{L}(\{\mathbf{x}\}, \{\mathbf{m}\}, \theta)$. Formally, $\mathcal{I}_\theta(\mathbf{x})$, i.e., the parameter perturbation if $\mathbf{x}$ was upweighted by ϵ, is given by*

$$\mathcal{I}_\theta(\mathbf{x}) \stackrel{def}{=} -\frac{d\hat{\theta}_{\epsilon,\mathbf{x}}}{d\epsilon}\bigg|_{\epsilon=0} = [\nabla\nabla\mathcal{L}(\mathbf{X}^0, \mathbf{M}^0, \hat{\theta}_0)]^{-1}\cdot\nabla\mathcal{L}(\{\mathbf{x}\}, \{\mathbf{m}\}, \hat{\theta}_0)$$

where $\nabla\nabla\mathcal{L}$ and $\nabla\mathcal{L}$ are the Hessian matrix and derivative of the imputation loss function $\mathcal{L}$ with respect to $\hat{\theta}_0$, respectively.

In Definition 2, the derivative $\nabla\mathcal{L}(\{\mathbf{x}\}, \{\mathbf{m}\}, \hat{\theta}_0)$ is given by

$$\nabla\mathcal{L}(\{\mathbf{x}\}, \{\mathbf{m}\}, \hat{\theta}_0) = \frac{\mathbf{m}\odot(\bar{\mathbf{x}} - \mathbf{x})\cdot\mathcal{D}(\mathbf{m})\cdot\nabla\bar{\mathbf{x}}}{||\mathbf{m}||_2^2}$$

where $\mathcal{D}(\mathbf{m})$ is to transform a vector $\mathbf{m}$ to a diagonal matrix, $\nabla\bar{\mathbf{x}}$ indicates the derivative of the vector $\bar{\mathbf{x}}$ with respect to $\hat{\theta}_0$. In addition,

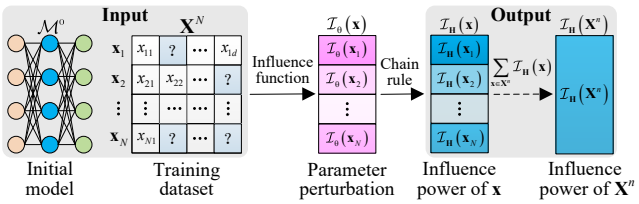


Figure 2: Illustration of the IIE module

we can write the Hessian matrix of $\mathcal{L}(\mathbf{X}^0, \mathbf{M}^0, \hat{\theta}_0)$ in the form

$$\begin{aligned}\nabla\nabla\mathcal{L}(\mathbf{X}^0, \mathbf{M}^0, \hat{\theta}_0) =& \frac{1}{N}\sum_{\mathbf{x}_i \in \mathbf{X}^0}\Bigg[\sum_{j=1}^{d}\frac{m_{ij}^2}{||\mathbf{m}_i||_2^2}\cdot(\bar{x}_{ij}-x_{ij})\cdot\nabla\nabla\bar{x}_{ij} \\ &+ \frac{[\mathcal{D}(\mathbf{m}_i)\cdot\nabla\bar{\mathbf{x}}_i]^\top\cdot\mathcal{D}(\mathbf{m}_i)\cdot\nabla\bar{\mathbf{x}}_i}{||\mathbf{m}_i||_2^2}\Bigg] \\ =& \frac{1}{N}\sum_{\mathbf{x}_i \in \mathbf{X}^0}\frac{[\mathcal{D}(\mathbf{m}_i)\cdot\nabla\bar{\mathbf{x}}_i]^\top\cdot\mathcal{D}(\mathbf{m}_i)\cdot\nabla\bar{\mathbf{x}}_i}{||\mathbf{m}_i||_2^2}\end{aligned}$$

The second equation follows by ignoring the first term in the first equation. The reason is that, the parametric imputation models [5, 45] ensure that, the vector $\bar{\mathbf{x}}_i$ predicted by the trained $\mathcal{M}^0$ happens to be very close to the observed values in $\mathbf{x}_i \in \mathbf{X}^0$.

In addition, for parametric deep learning imputation methods, the Hessian matrix $\nabla\nabla\mathcal{L}(\mathbf{X}^0, \mathbf{M}^0, \hat{\theta}_0)$ may be singular and its inverse does not exist in many cases. To this end, we add a simple term $\lambda\mathbf{I}$ to $\nabla\nabla\mathcal{L}(\mathbf{X}^0, \mathbf{M}^0, \hat{\theta}_0)$, where λ is a small positive quantity. Thus, the IIE module actually finds the inverse of $\nabla\nabla\mathcal{L}(\mathbf{X}^0, \mathbf{M}^0, \hat{\theta}_0) + \lambda\mathbf{I}$, which is completely invertible. In this way, $\mathcal{I}_\theta(\mathbf{x})$ is calculated by $\mathcal{I}_\theta(\mathbf{x}) = [\nabla\nabla\mathcal{L}(\mathbf{X}^0, \mathbf{M}^0, \hat{\theta}_0) + \lambda\mathbf{I}]^{-1}\cdot\nabla\mathcal{L}(\{\mathbf{x}\}, \{\mathbf{m}\}, \hat{\theta}_0)$.

Then, we employ the chain rule [20] to approximate the change in the imputation loss when adding/deleting $\mathbf{x}$ into/from $\mathbf{X}^0$, termed as the *influence power* $\mathcal{I}_\mathbf{H}(\mathbf{x})$, as stated in Definition 3.

Definition 3. **(Influence power)**. *Given a sample* $\mathbf{x}$ *with its mask vector* $\mathbf{m}$*, the influence power of* $\mathbf{x}$*, denoted by* $\mathcal{I}_\mathbf{H}(\mathbf{x})$*, indicates the change in the imputation loss of* $\mathcal{M}^0$ *for the holdout dataset* $\mathbf{H}$ *when all observed values in* $\mathbf{x}$ *are upweighted. It can be formally approximated by the chain rule.*

$$\mathcal{I}_\mathbf{H}(\mathbf{x}) \overset{def}{=} -\frac{d\mathcal{L}\left(\mathbf{H}, \mathbf{M}^h, \hat{\theta}_{\epsilon,\mathbf{x}}\right)}{d\epsilon}\Bigg|_{\epsilon=0} = \nabla\mathcal{L}\left(\mathbf{H}, \mathbf{M}^h, \hat{\theta}_0\right)^\top\cdot\mathcal{I}_\theta(\mathbf{x})$$

In other words, the sample $\mathbf{x}$ with the positive influence power has positive effect on the imputation model's prediction, otherwise $\mathbf{x}$ has negative effect. Besides, due to the additivity of the influence function [19, 29], both the parameter perturbation and influence power preserve the additive property. As a result, the parameter perturbation or the influence power of a sample set $\mathbf{X}^n$ is the sum of the corresponding values of $\mathbf{x} \in \mathbf{X}^n$, i.e., $\mathcal{I}_\theta(\mathbf{X}^n) = \sum_{\mathbf{x}\in\mathbf{X}^n}\mathcal{I}_\theta(\mathbf{x})$ and $\mathcal{I}_\mathbf{H}(\mathbf{X}^n) = \sum_{\mathbf{x}\in\mathbf{X}^n}\mathcal{I}_\mathbf{H}(\mathbf{x})$.

The IIE module is finally depicted in Figure 2. It takes the initial model $\mathcal{M}^0$ and the training dataset $\mathbf{X}^N$ as inputs, and outputs the influence powers of samples $\mathbf{x} \in \mathbf{X}^N$. Specifically, IIE first employs the influence function to estimate the parameter perturbation by upweighting the observed values in $\mathbf{x}$ with an infinitesimal amount, when adding/deleting $\mathbf{x}$ into/from $\mathbf{X}^0$. It then utilizes the chain rule to further infer the change in the imputation loss of $\mathcal{M}^0$, i.e., the $\mathbf{x}$'s influence power. Moreover, for a sample set $\mathbf{X}^n$, its influence power is the sum of the influence powers of all samples in $\mathbf{X}^n$.

5 RSS MODULE

In this section, we first explain how to estimate the *imputation accuracy* using the influence power. Then, we detail the construction of a *minimum representative set* $\mathbf{X}^*$ in RSS under accuracy guarantees. We also present an effective *weighted* imputation loss function to further boost the performance of parametric models.

5.1 Imputation Accuracy Estimation

The imputation accuracy estimation is an indispensable part of the RSS module to select the most beneficial samples for training, in order to satisfy imputation accuracy guarantee. It also helps to avoid retraining the model, thus improves the efficiency.

Specifically, in the RSS module, based on the influence power tailored to (in)complete samples, we can approximately estimate the imputation loss (i.e., accuracy) of the initial model $\mathcal{M}^0$ for the holdout dataset $\mathbf{H}$, when adding a sample set $\mathbf{X}^n \subseteq \mathbf{X}^c$ into the initial set $\mathbf{X}^0$. In particular, the influence power $\mathcal{I}_\mathbf{H}(\mathbf{X}^n)$ of $\mathbf{X}^n$ indicates the change in the imputation loss of $\mathcal{M}^0$ for $\mathbf{H}$, when adding $\mathbf{X}^n$ into $\mathbf{X}^0$. In other words, with the imputation loss $\mathcal{L}(\mathbf{H}, \mathbf{M}^h, \hat{\theta}_0)$ of $\mathcal{M}^0$ for $\mathbf{H}$ determined by Eq. 2, we can analytically infer the specific imputation loss of $\mathcal{M}^0$ for $\mathbf{H}$ if adding $\mathbf{X}^n$ into $\mathbf{X}^0$.

To be more specific, adding $\mathbf{X}^n$ into $\mathbf{X}^0$ is the same as upweighting $\mathbf{X}^n$ by $\epsilon = \frac{1}{N}$ in $\hat{\theta}_{\epsilon,\mathbf{x}}$. Hence, according to Definition 2, we can linearly approximate the parameter perturbation for adding $\mathbf{X}^n$ into $\mathbf{X}^0$ by computing $\hat{\theta}_0 - \hat{\theta}_{+\mathbf{X}^n} = \frac{1}{N}\sum_{\mathbf{x}\in\mathbf{X}^n}\mathcal{I}_\theta(\mathbf{x})$. With the chain rule, the change in the imputation loss is given by $\mathcal{L}(\mathbf{H}, \mathbf{M}^h, \hat{\theta}_0) - \hat{\mathcal{L}}_\mathbf{H}(\mathbf{X}^n) = \frac{1}{N}\sum_{\mathbf{x}\in\mathbf{X}^n}\mathcal{I}_\mathbf{H}(\mathbf{x})$. As a result, the imputation loss $\hat{\mathcal{L}}_\mathbf{H}(\mathbf{X}^n)$ with adding $\mathbf{X}^n$ into $\mathbf{X}^0$ can be estimated by

$$\hat{\mathcal{L}}_\mathbf{H}(\mathbf{X}^n) = \mathcal{L}(\mathbf{H}, \mathbf{M}^h, \hat{\theta}_0) - \frac{1}{N}\mathcal{I}_\mathbf{H}(\mathbf{X}^n) \tag{3}$$

According to Eq. 3, one can *directly* derive the imputation loss/accuracy after adding more training samples $\mathbf{X}^n$, without retraining the imputation model, where the imputation loss $\mathcal{L}(\mathbf{H}, \mathbf{M}^h, \hat{\theta}_0)$ of $\mathcal{M}^0$ for $\mathbf{H}$ can be known in advance, as well as the influence power $\frac{1}{N}\mathcal{I}_\mathbf{H}(\mathbf{X}^n)$. It is also easy to realize that, the imputation accuracy estimation using Eq. 3 is *accurate*.

In addition, when we add the whole candidate dataset $\mathbf{X}^c$ to $\mathbf{X}^0$, the imputation loss $\hat{\mathcal{L}}_\mathbf{H}(\mathbf{X}^c)$ with the training dataset $\mathbf{X}^N = \mathbf{X}^c \cup \mathbf{X}^0$ can be estimated by $\hat{\mathcal{L}}_\mathbf{H}(\mathbf{X}^c) = \mathcal{L}(\mathbf{H}, \mathbf{M}^h, \hat{\theta}_0) - \frac{1}{N}\sum_{\mathbf{x}\in\mathbf{X}^c}\mathcal{I}_\mathbf{H}(\mathbf{x})$.

Moreover, we allow users to tolerate the imputation accuracy with a user-specified hype-parameter α. Formally, the user-specified imputation loss can be estimated by $\hat{\mathcal{L}}^\alpha_\mathbf{H}(\mathbf{X}^c) \overset{\text{def}}{=} \frac{1}{\alpha}\cdot\hat{\mathcal{L}}_\mathbf{H}(\mathbf{X}^c)$. In other words, for the imputation accuracy $\hat{\mathcal{L}}_\mathbf{H}(\mathbf{X}^c)$ (i.e., the highest accuracy that can be derived using the training data $\mathbf{X}^N = \mathbf{X}^c \cup \mathbf{X}^0$), the users can flexibly set the value of α, so as to get a satisfactory imputation accuracy (over the holdout set $\mathbf{H}$) that is not worse than $\frac{1}{\alpha}\cdot\hat{\mathcal{L}}_\mathbf{H}(\mathbf{X}^c)$. Note that, the smaller the α value, the higher the user-specified imputation loss $\hat{\mathcal{L}}^\alpha_\mathbf{H}(\mathbf{X}^c)$, the fewer the training

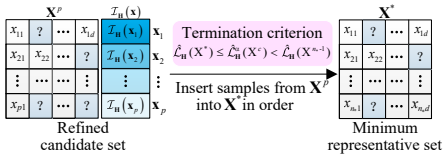


Figure 3: Illustration of the RSS module

samples requested for ensuring model accuracy, and thus the less the imputation model training cost.

5.2 Sample Selection with Accuracy Guarantee

The RSS module is responsible for seeking the minimum representative sample set $\mathbf{X}^*$ with the user-specific accuracy tolerance, i.e., the estimated imputation loss $\hat{\mathcal{L}}_{\mathbf{H}}(\mathbf{X}^*)$ should not be worse than the user-specified imputation loss $\hat{\mathcal{L}}^{\alpha}_{\mathbf{H}}(\mathbf{X}^c)$.

First, it is obvious that, the higher the influence power of the sample set $\mathbf{X}^*$, the smaller the estimated loss $\hat{\mathcal{L}}_{\mathbf{H}}(\mathbf{X}^*)$, according to Eq. 3. In other words, if samples with higher influence powers are included in $\mathbf{X}^*$, the fewer samples (in $\mathbf{X}^*$) are requested for satisfying the user-specified imputation loss $\hat{\mathcal{L}}^{\alpha}_{\mathbf{H}}(\mathbf{X}^c)$. To this end, we find the highest influence power samples from $\mathbf{X}^c$ to form the minimum representative set $\mathbf{X}^*$ for the user-specific accuracy tolerance.

Figure 3 illustrates the general procedure of the RSS module to find the minimum representative sample set $\mathbf{X}^*$. Specifically, we first get a *refined* candidate set $\mathbf{X}^p$ containing the samples in $\mathbf{X}^c$ with the *positive* influence powers. We sort these samples in $\mathbf{X}^p$ in the non-ascending order of the influence power. Then, we collect the samples from $\mathbf{X}^p$ that have the highest influence powers to form the *minimum representative sample set* $\mathbf{X}^*$. The collection procedure terminates if the estimated loss $\hat{\mathcal{L}}_{\mathbf{H}}(\mathbf{X}^*)$ is equal to or smaller than $\hat{\mathcal{L}}^{\alpha}_{\mathbf{H}}(\mathbf{X}^c)$ while $\hat{\mathcal{L}}_{\mathbf{H}}(\mathbf{X}^{n_*-1})$ is higher than $\hat{\mathcal{L}}^{\alpha}_{\mathbf{H}}(\mathbf{X}^c)$, i.e., $\hat{\mathcal{L}}_{\mathbf{H}}(\mathbf{X}^*) \leq \hat{\mathcal{L}}^{\alpha}_{\mathbf{H}}(\mathbf{X}^c) < \hat{\mathcal{L}}_{\mathbf{H}}(\mathbf{X}^{n_*-1})$, where $\mathbf{X}^{n_*-1}$ consists of the samples with the top $(n_* - 1)$ influence powers in $\mathbf{X}^c$. Otherwise, all samples in $\mathbf{X}^p$ are inserted in $\mathbf{X}^*$ (i.e., $\mathbf{X}^* = \mathbf{X}^p$).

5.3 Training with Weighted Samples

We also leverage the concept of influence power to boost the parametric imputation model performance in the EDIT system.

In the standard training, the imputation model aims to minimize the (basic) imputation loss function defined in Eq. 2, where each training sample is equally weighted. Actually, different samples have different influence powers on the imputation models, based on Definition 3. The higher the influence power of the (in)complete sample $\mathbf{x}$, the stronger the effect of $\mathbf{x}$ on the prediction result of the parametric imputation model when adding $\mathbf{x}$ to the training data.

In view of this, to obtain an optimized parametric imputation model $\mathcal{M}^*$ by EDIT, we develop an *effective weighted* imputation loss function $\mathcal{L}_f$ that weights the training samples with the influence power on top of the basic loss function in Eq. 2, i.e.,

$$\mathcal{L}_f(\mathbf{X}^f, \mathbf{M}^f, \theta) = \frac{1}{n_f} \sum_{i=1}^{n_f} \mathcal{I}_{\mathbf{H}}(\mathbf{x}_i) \frac{||\mathbf{m}_i \odot (\bar{\mathbf{x}}_i - \mathbf{x}_i)||_2^2}{2||\mathbf{m}_i||_2^2}$$

where $\mathbf{X}^f = \mathbf{X}^0 \cup \mathbf{X}^*$, $\mathbf{M}^f = \mathbf{M}^0 \cup \mathbf{M}^*$, and $n_f = n_0 + n_*$. The new loss function $\mathcal{L}_f$ drives the imputation model to pay more attention on the samples with higher influence powers. One can observe that, the RSS module eliminates or downweights the samples with the low or even negative influence powers during model training via the weighted imputation loss function. It thus facilitates the higher robustness of imputation models against the dirty data.

6 EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of our proposed system EDIT with ten state-of-the-art parametric imputation methods. All algorithms were implemented in Python. The experiments were conducted on an Intel Core 2.80GHz server with TITAN Xp 12GiB (GPU) and 192GB RAM, running Ubuntu 18.04 system.

Datasets. In the experiments, we use four real-world large-scale datasets: (i) ***Power*** [13] contains 1,045,295 samples with 114 features gathered in a house located in Sceaux between December 2006 and November 2010. (ii) ***Gas*** [17] includes 4,178,504 chemical sensor reading records exposed to gas mixtures at 56 concentration levels. (iii) ***HIGGS*** [10] contains 11,000,000 samples with 28 features. Each sample is a pair of physical properties of an environment and a binary indicator of Higgs bosons production. (iv) ***Criteo*** [9] is a click-through rates dataset with 45,840,616 samples and 39 features. It is about display advertisement made publicly available by Criteo Labs. For all datasets, we randomly choose 10% data as the test data, 1% data as the holdout data, and the rest as the training data.

Metrics. In the evaluation, we use the *training time* and *root mean square error* (RMSE) to measure the efficiency and effectiveness of imputation models. We also report the *training sample rate* R_t (i.e., how many samples are used for training imputation model) of imputation models. For EDIT, R_t is $\frac{n_f}{N} \times 100\%$, and its training time includes influence computation, sample selection, and model training times. All metrics follow the rule that, the smaller the metric value, the better efficiency or effectiveness the imputation algorithms. To obtain the RMSE values, we remove 50% observed values under the missing mechanisms. We use them as the ground-truth to calculate the RMSE values during test. In our evaluation, each value is reported by averaging five times of results under different data random divisions.

Missing mechanisms. In the experiments, we inspect three missing mechanisms, i.e., missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR).

First, the data missingness in **MCAR** [43] is unrelated to any values. The missing values are randomly injected to $\mathbf{X}$, in order to simulate the default missing mechanism MCAR. The data missingness in **MAR** is only related to the observed data. We follow the methodology in [8] to simulate MAR. It first selects a numerical attribute f_v. For a sample $\mathbf{x}_i$, the value x_{ij} in a feature f_j is missing at the probability of $\mathrm{P}_c(x_{ij})$, which is calculated according to the value (denoted by x_{iv}) of $\mathbf{x}_i$ on f_v. Namely, $\mathrm{P}_c(x_{ij}) = \frac{\Phi(x_{iv})}{\sum_{i=1}^{s} \Phi(x_{iv})}$, where $\Phi(x_{iv})$ denotes the ranking of x_{iv} in f_v. The smaller the value in f_v, the higher the ranking. By contrast, the data missingness in **MNAR** is only related to the missing values themselves. Similar as [37], to simulate MNAR, the value x_{im} of a sample $\mathbf{x}_i$ is missing at the probability of $\mathrm{P}_m(x_{im})$, i.e., $\mathrm{P}_m(x_{im}) = \frac{\Phi(x_{im})}{\sum_{i=1}^{s} \Phi(x_{im})}$.

Imputation methods. In the experiments, the baseline imputation methods include ten state-of-the-art parametric imputation algorithms: three machine learning ones (i.e., MissF, Baran, and

Table 2: Imputation performance comparison on *Power*

Method		RMSE (Bias)	Time (s)	R_t (%)
MissF	Original	–	–	–
	EDIT	0.240 (± 0.060)	34,362	15.63
Baran	Original	–	–	–
	EDIT	0.212 (± 0.022)	96,261	8.98
MICE	Original	–	–	–
	EDIT	0.233 (± 0.029)	28,046	14.90
DataWig	Original	–	–	–
	EDIT	0.182 (± 0.021)	24,231	6.87
RRSI	Original	–	–	–
	EDIT	0.158 (± 0.014)	18,123	**2.12**
MIDAE	Original	–	–	–
	EDIT	0.218 (± 0.041)	21,012	10.26
VAEI	Original	–	–	–
	EDIT	0.233 (± 0.015)	12,682	**2.12**
HIVAE	Original	0.119 (± 0.056)	3,504	100
	EDIT	**0.116 (± 0.024)**	**1,573**	3.88
GINN	Original	–	–	–
	EDIT	0.220 (± 0.098)	22,621	5.34
GAIN	Original	0.183 (± 0.038)	2,909	100
	EDIT	0.163 (± 0.042)	1,794	8.26

MICE) and seven deep learning ones (i.e., DataWig, RRSI, MIDAE, VAEI, HIVAE, GINN, and GAIN). We evaluate the proposed EDIT system on top of the ten models, compared with these basic parametric imputation algorithms (i.e., the *original* ones).

Implementation details. For all machine learning algorithms, the learning rate is set to 0.3, and the number of iterations is set to 100. In particular, the number of decision trees in MissFI is set to 6. Baran employs AdaBoost as the prediction model. The imputation times in MICE are 20. For all deep learning imputation methods, the learning rate is 0.001, the dropout rate is 0.5, the training epoch is 30, and the batch size is 128. The ADAM algorithm is utilized to train networks. MIDAE is a 2-layer with 128 units per layer network. For VAEI, the encoder and decoder are fully connected networks with two hidden layers, each with 20 neurons per layer, and the latent space was 10-dimensional. HIVAE uses only one dense layer for all the parameters of the encoder and decoder, each with 10 neurons per layer. In GINN, the discriminator used is a simple 3-layer feed-forward network trained 5 times for each optimization step of the generator. In GAIN, both generator and discriminator are modeled as 2-layer fully connected network. For EDIT, the hype-parameter α is by default set to 1. The initial sample size n_0 is 6,000 for *Power*, 10,000 for *Gas*, 20,000 for *HIGGS*, and 40,000 for *Criteo*.

6.1 Scalability Evaluation

The first set of experiments explores the performance of the proposed EDIT system over existing parametric imputation methods. The experimental results are reported in Table 2 and Table 3. Some results are unavailable, since the corresponding methods are not able to finish within 10^5 seconds.

One can observe that, compared to each original method, EDIT takes less training time and samples (i.e., smaller training sample rate R_t), while it achieves a higher imputation accuracy (i.e. smaller RMSE value). This is because that, the RSS module with the representative sample set minimizes the required training samples of parametric models to satisfy the imputation accuracy expectation of users, and thereby saving the training time. In addition, EDIT utilizes the influence power to present an effective weighted imputation loss function for accuracy enhancement. Specifically, EDIT spends only 48.06% of the training time required for original models in average, and it decreases to 21.28% for HIVAE on *Gas* dataset. Meanwhile, EDIT adopts only 6.56% of the training samples used for original models in average. The training sample rate even decreases to 2.12% for VAEI and RRSI on *Power* dataset. In terms of RMSE, EDIT exceeds the corresponding original methods by 3.36% in average (even 11.62% for GAIN on *Gas*). Note that, in the rest of experiments, we employ HIVAE and GAIN as baselines to demonstrate the performance of EDIT, since they can get the original imputation results and thus provide a clear comparison benchmark to reflect the superiority of EDIT.

Moreover, we investigate the effectiveness of the weighted imputation loss function adopted in EDIT, as well as the strategy of using the samples with the high positive influence powers. We compare EDIT with three methods, i.e., *noW-EDIT* that trains the final imputation model with original imputation loss function, i.e., Eq. 2, *noW-EDIT-* that selects top-R_t samples with *negative* influence powers on top of noW-EDIT, and *Random* that randomly chooses R_t samples to add the training data. The training sample rate R_t is determined by EDIT, as reported in Table 2 and Table 3.

The corresponding experimental results are shown in Table 4. First, EDIT consistently gets the better imputation accuracy than noW-EDIT over the four datasets. It is attributed to the effectiveness of the weighted imputation loss function used in EDIT. noW-EDIT exceeds Random and noW-EDIT- by 4.72% and 7.83% in average, respectively. Random performs better than noW-EDIT- in all cases. We can conclude that, the samples with positive/negative influence powers do have positive/negative effect on the imputation model training. In other words, using the samples with the positive influence powers indeed benefits the imputation model predictions.

6.2 Parameter Evaluation

Effect of missing rate. When varying the missing rate R_m (i.e., how many values in original observed data are dropped) from 10% to 90%, the corresponding results of the RMSE, training time, and R_t are depicted in Figure 4. We also report the time cost of IIE, which is the core module of EDIT. In Figure 4 and the rest of experiments, the results of HIVAE and GAIN are unavailable over *Criteo*, since they are not able to finish within 10^5 seconds.

We can find that, compared with original methods, EDIT takes much less training time and samples to obtain a better imputation accuracy in all cases. It is robust with the increasing missing rate R_m, due to the slight drop in RMSE. In EDIT, the IIE module takes 84.26% training time in average. It indicates that, calculating the influence power of samples in $\mathbf{X}^N$ takes the dominant cost. Furthermore, with the growth of R_m, the imputation accuracy descends consistently for each algorithm. The reason is that, the data information turns less with the increase of R_m, resulting in a lower accuracy.

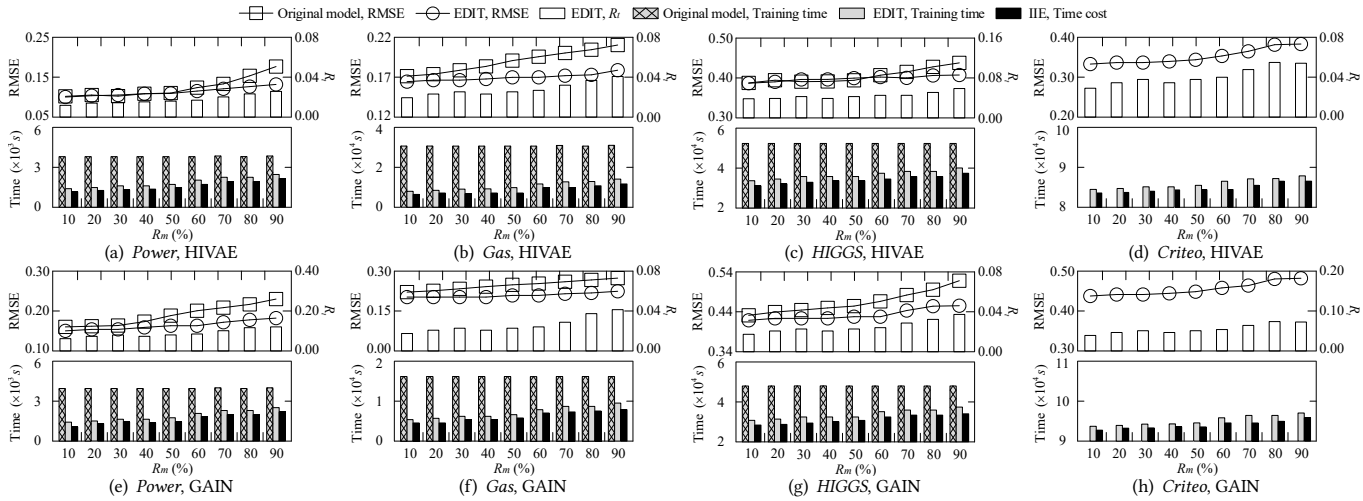
Effect of α. With α varying from 80% to 120%, Figure 5 plots the corresponding experimental results, including the imputation loss values over the holdout dataset $\mathbf{H}$, the sample rate of the refined candidate set $\mathbf{X}^p$ (the minimum representative sample set $\mathbf{X}^*$), i.e.,

Table 3: Imputation performance comparison on *Gas*, *HIGGS*, and *Criteo*

Method		*Gas*			*HIGGS*			*Criteo*		
		RMSE (Bias)	Time (s)	R_t (%)	RMSE (Bias)	Time (s)	R_t (%)	RMSE (Bias)	Time (s)	R_t (%)
HIVAE	Original	0.182 (± 0.089)	32,291	100	**0.404 (± 0.031)**	54,721	100	–	–	–
	EDIT	**0.169 (± 0.067)**	6,872	2.81	0.412 (± 0.063)	35,621	4.22	**0.324** (± 0.063)	94,291	**3.96**
GAIN	Original	0.241 (± 0.037)	17,548	100	0.448 (± 0.092)	44,044	100	–	–	–
	EDIT	0.213 (± 0.036)	**6,199**	**2.43**	0.423 (± 0.125)	**29,443**	**2.91**	0.366 (± 0.045)	**85,683**	4.22

Table 4: Imputation performance comparison (RMSE) of Random, noW-EDIT−, and noW-EDIT

Method		*Power*	*Gas*	*HIGGS*	*Criteo*
HIVAE	Random	0.141 (± 0.034)	0.187 (± 0.058)	0.433 (± 0.064)	0.345 (± 0.070)
	noW-EDIT-	0.149 (± 0.042)	0.194 (± 0.067)	0.444 (± 0.058)	0.359 (± 0.061)
	noW-EDIT	**0.120 (± 0.024)**	**0.175 (± 0.060)**	**0.420 (± 0.063)**	**0.329 (± 0.063)**
GAIN	Random	0.184 (± 0.038)	0.243 (± 0.045)	0.452 (± 0.094)	0.386 (± 0.032)
	noW-EDIT-	0.192 (± 0.043)	0.252 (± 0.032)	0.460 (± 0.099)	0.397 (± 0.038)
	noW-EDIT	**0.173 (± 0.042)**	**0.221 (± 0.036)**	**0.425 (± 0.102)**	**0.375 (± 0.045)**



(a) *Power*, HIVAE (b) *Gas*, HIVAE (c) *HIGGS*, HIVAE (d) *Criteo*, HIVAE

(e) *Power*, GAIN (f) *Gas*, GAIN (g) *HIGGS*, GAIN (h) *Criteo*, GAIN

Figure 4: Imputation performance comparison under different missing rates

$R_1 = n_p/N$ ($R_2 = n_*/N$). The user-specified loss value is derived by $\frac{1}{\alpha} \cdot \mathcal{L}(\mathbf{H}, \hat{\theta})$, where $\mathcal{L}(\mathbf{H}, \hat{\theta})$ is the loss of EDIT trained on $\mathbf{X}^N$.

As inferred from the figure, the imputation loss values of EDIT and noW-EDIT are smaller than (or approximate) the user-specified loss in all cases. It means that, EDIT and noW-EDIT get at least as a good accuracy as the user's expectation, i.e., they indeed satisfy the accuracy requirement of users. Besides, EDIT obtains the lower imputation loss than noW-EDIT in each case. It confirms the power of the newly presented weighted imputation loss function for EDIT.

Moreover, the loss values of EDIT and noW-EDIT decrease with the growth of α, while the sample rate R_2 increases in most cases. It is because, a larger value of α signifies a smaller user-specified loss (w.r.t. a higher accuracy expectation of users). Hence, more training samples (i.e., a larger R_2) are needed to satisfy the higher accuracy requirement, resulting in the lower imputation loss. In addition, R_1 is constant, since α does not influence the number of samples in $\mathbf{X}^c$ that have positive influence powers (i.e., the refined candidate set $\mathbf{X}^p$ is fixed for varying α). Especially, when α is smaller than 100%, the imputation loss of EDIT changes fast, while the sample rate R_2 is the opposite. When α exceeds 100%, R_2 obviously increases to satisfy the accuracy guarantee. When α exceeds 110%, the imputation loss of EDIT changes slightly, where the newly added samples in $\mathbf{X}^*$ for training are fully included in $\mathbf{X}^p$ (containing the *positive* influence power samples in $\mathbf{X}^c$). Thus, if users want to maximize the efficiency of EDIT, they can choose the smaller α value (e.g., less than 100%). If they would like to maximize the effectiveness of EDIT, the large α value (even greater than 100%) can be chosen.

Effect of missing mechanisms. We vary the data missing mechanisms MCAR, MAR, and MNAR, and the corresponding experimental results are plotted in Figure 6.

We can observe that, the execution time of imputation algorithms HIVAE, GAIN, and EDIT is insensitive to different missing mechanisms for each dataset. The imputation algorithms in MNAR mechanism achieve lower accuracy than that in other two missing mechanisms. It partially attributes to the extreme biased simulation of MNAR. In contrast, the performance of the biased missing case in MAR is similar as that in MCAR. Thus, the study of missing value distributions is more complicated than expected. It is worth further exploration in the future. In addition, as expected, EDIT consistently takes much less training time to obtain a better imputation accuracy in each case. It spends 52.58%, 52.47%, and 56.83% of the training time required for original models to achieve 5.14%, 3.55%, and 2.53% accuracy gain in average under MCAR, MAR, and MNAR mechanisms, respectively.

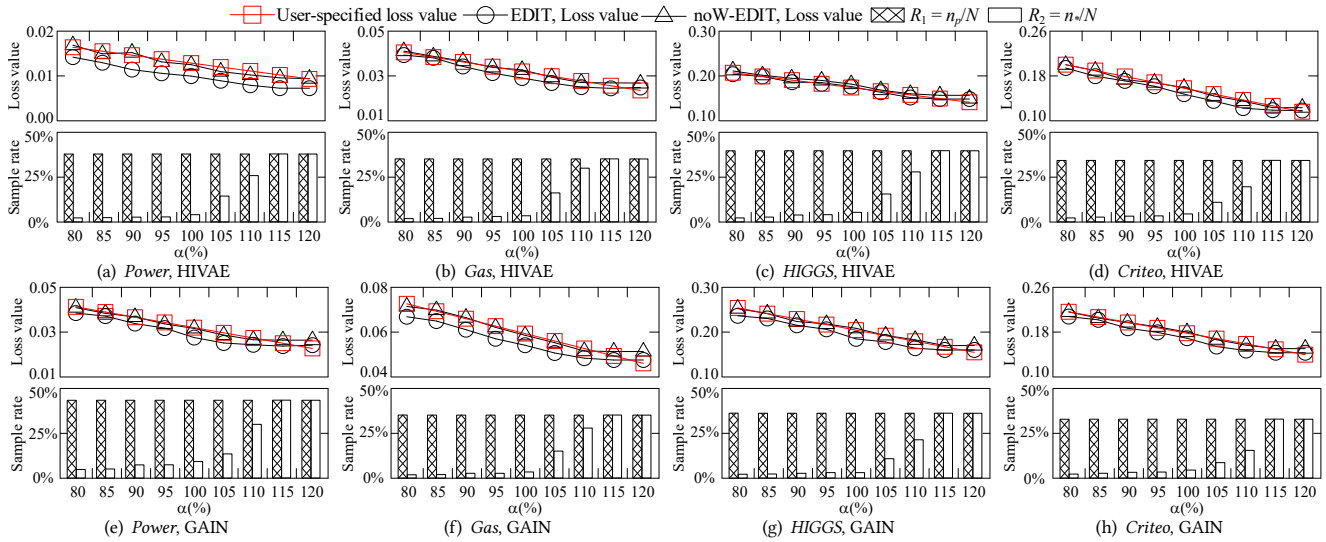


Figure 5: Imputation performance comparison under different values of α

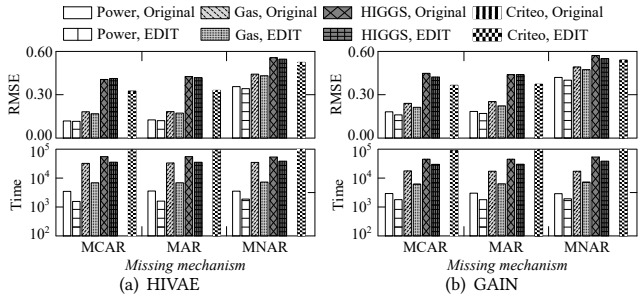


Figure 6: Comparison under different missing mechanisms

Table 5: Imputation evaluation over *Mobility*

Metric	HIVAE	EDIT-HIVAE	GAIN	EDIT-GAIN
Time (s)	9,074	5,988	6,533	**5,134**
R_t (%)	100	**5.14**	100	10.42

Table 6: Post-imputation evaluation (MAE) over *Mobility*

HIVAE	EDIT-HIVAE	GAIN	EDIT-GAIN
106.52(±12.26)	**100.26(±8.89)**	111.71(±13.45)	102.26(±12.02)

6.3 Prediction on Real-world Mobility Data

The last set of experiments verifies the superiority of EDIT to original imputation methods on *post-imputation prediction* task.

We employ a real-world *incomplete* COVID-19 community mobility (i.e., *Mobility*) dataset [39]. It shows how length of stay at different places change compared to a baseline in a specific region (totally 7,550 regions with 2,268,105 samples, taking 30.62% average missing rate). The regression task is to predict the count of new cases confirmed after a positive test. In particular, the imputation methods are first employed to impute missing values in *Mobility*. Then, a regression model is trained with three fully connected layers over the imputed data. The training epoch is 30 and the learning rate is 0.005. The initial sample size in EDIT is 10,000.

First, the imputation performances of the training time and training sample rate R_t over *Mobility* are presented in Table 5. We can find that, the EDIT-based methods take only 71.26% training time and 7.78% training samples required for original imputation models in average. Thus, it further confirms the efficiency of the EDIT system. Moreover, the post-imputation prediction results are depicted in Table 6 over *Mobility*. The smaller *mean absolute error* (MAE) corresponds to the better prediction effect. We can further observe that, the prediction performance under different imputation algorithms is consistent with the imputation performance of these algorithms, i.e., the EDIT ones have better imputation accuracy than original ones. In particular, EDIT exceeds the corresponding original method by 6.96% in average, and it increases up to 8.46% for GAIN.

7 CONCLUSIONS

In this paper, we propose an efficient and effective data imputation system EDIT with the influence functions. EDIT consists of an IIE module and a RSS module. The RSS module finds a *minimum representative sample* set using the concept of *influence power* introduced by the IIE module, so that the model training theoretically satisfies the user-specified imputation accuracy. We also present an effective weighted imputation loss function to further boost imputation performance. Extensive experiments over several real-world datasets demonstrate that, EDIT is able to significantly accelerate the model training while having the obvious accuracy gain, compared with ten state-of-the-art parametric imputation methods.

ACKNOWLEDGMENTS

This work is partly supported by the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under Grant No.LR21F020005, the National Natural Science Foundation of China under Grants No.61902343, No.62025206, No.61972338, No.62102351, No.61825205, the National Key Research and Development Program of China under Grant No.2019YFE0126200, the National Science and Technology Major Project of China under Grant No.50-D36B02-9002-16/19, and the Fundamental Research Funds for the Central Universities under Grant No.2021FZZX001-25. Jianwei Yin is the corresponding author of the work.

REFERENCES

[1] Naomi S Altman. 1992. An introduction to kernel and nearest -neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *ICML*. 214–223.

[3] Laure Berti-Equille, Hazar Harmouch, Felix Naumann, Noël Novelli, and Thirumuruganathan Saravanan. 2018. Discovery of genuine functional dependencies from relational data with missing values. *Proceedings of the VLDB Endowment* 11, 8 (2018), 880–892.

[4] Felix Biessmann, Tammo Rukat, Philipp Schmidt, Prathik Naidu, Sebastian Schelter, Andrey Taptunov, Dustin Lange, and David Salinas. 2019. DataWig: Missing value imputation for tables. *Journal of Machine Learning Research* 20, 1 (2019), 1–6.

[5] Muzellec Boris, Josse Julie, Boyer Claire, and Cuturi Marco. 2020. Missing data imputation using optimal transport. In *ICML*. 1–18.

[6] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2015. Importance weighted autoencoders. *ArXiv Preprint ArXiv:1509.00519* (2015).

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *SIGKDD*. 785–794.

[8] Adriana Fonseca Costa, Miriam Seoane Santos, Jastin Pompeu Soares, and Pedro Henriques Abreu. 2018. Missing data imputation via denoising autoencoders: The untold story. In *IDA*. 87–98.

[9] CriteoLabs. 2014. http://labs.criteo.com/2014/02/ download-kaggle-display-advertising-challenge-dataset/. (2014).

[10] Whiteson Daniel. 2014. https://archive.ics.uci.edu/ml/datasets/HIGGS. (2014).

[11] Guowang Du, Lihua Zhou, Yudi Yang, Kevin Lü, and Lizhen Wang. 2021. Deep multiple auto-encoder-based multi-view clustering. *Data Science and Engineering* 6, 1 (2021), 1–16.

[12] Alireza Farhangfar, Lukasz A Kurgan, and Witold Pedrycz. 2007. A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man, and Cybernetics -Part A: Systems and Humans* 37, 5 (2007), 692–709.

[13] Hebrail Georges and Berard Alice. 2012. https://archive.ics.uci.edu/ml /datasets/Individual+household+electric+power+consumption. (2012).

[14] Lovedeep Gondara and Ke Wang. 2017. Multiple imputation using deep denoising autoencoders. *ArXiv Preprint ArXiv:1705.02737* (2017).

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NeurIPS*. 2672–2680.

[16] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.

[17] Burgu Javier. 2019. https://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+temperature+modulation. (2019).

[18] José M Jerez, Ignacio Molina, Pedro J García-Laencina, Emilio Alba, Nuria Ribelles, Miguel Martín, and Leonardo Franco. 2010. Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial Intelligence in Medicine* 50, 2 (2010), 105–115.

[19] Pang Wei Koh, Kai-Siang Ang, Hubert HK Teo, and Percy Liang. 2019. On the accuracy of influence functions for measuring group effects. *ArXiv Preprint ArXiv:1905.13289* (2019).

[20] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. *ArXiv Preprint ArXiv:1703.04730* (2017).

[21] Tongyu Liu, Ju Fan, Yinqing Luo, Nan Tang, Guoliang Li, and Xiaoyong Du. 2021. Adaptive data augmentation for supervised learning over missing data. *Proceedings of the VLDB Endowment* 14, 7 (2021), 1202–1214.

[22] Rhea Mahajan and Vibhakar Mansotra. 2021. Predicting geolocation of tweets: Using combination of CNN and BiLSTM. *Data Science and Engineering* 6, 4 (2021), 402–410.

[23] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective error correction via a unified context representation and transfer learning. *Proceedings of the VLDB Endowment* 13, 12 (2020), 1948–1961.

[24] Pierre-Alexandre Mattei and Jes Frellsen. 2019. MIWAE: Deep generative modelling and imputation of incomplete data sets. In *ICML*. 4413–4423.

[25] John T McCoy, Steve Kroon, and Lidia Auret. 2018. Variational autoencoders for missing data imputation with application to a simulated milling circuit. *IFAC-PapersOnLine* 51, 21 (2018), 141–146.

[26] Xiaoye Miao, Yangyang Wu, Jun Wang, Yunjun Gao, Xudong Mao, and Jianwei Yin. 2021. Generative semi-supervised learning for multivariate time series imputation. In *AAAI*. 8983–8991.

[27] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. 2018. Handling incomplete heterogeneous data using VAEs. *ArXiv Preprint ArXiv:1807.03653* (2018).

[28] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. 2017. HoloClean: Holistic data repairs with probabilistic inference. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1190–1201.

[29] Zhongzheng Ren, Raymond A. Yeh, and Alexander G. Schwing. 2020. Not all unlabeled data are equal: Learning to weight data in semi-supervised learning. In *NeurIPS*. 1–12.

[30] Patrick Royston and Ian R White. 2011. Multiple imputation by chained equations (MICE): Implementation in Stata. *Journal of Statistical Software* 45, 4 (2011), 1–20.

[31] Donald B Rubin. 1976. Inference and missing data. *Biometrika* 63, 3 (1976), 581–592.

[32] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *ArXiv Preprint ArXiv:1609.04747* (2016).

[33] Ognjen Savković, Paramita Mirza, Alex Tomasi, and Werner Nutt. 2013. Complete approximations of incomplete queries. *Proceedings of the VLDB Endowment* 6, 12 (2013), 1378–1381.

[34] Mohamed A Soliman, Ihab F Ilyas, and Shalev Ben-David. 2010. Supporting ranking queries on uncertain and incomplete data. *The VLDB Journal* 19, 4 (2010), 477–501.

[35] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. 2019. Missing data imputation with adversarially-trained graph convolutional networks. *ArXiv Preprint ArXiv:1905.01907* (2019).

[36] Daniel J Stekhoven and Peter Bühlmann. 2011. MissForest non parametric missing value imputation for mixed-type data. *Bioinformatics* 28, 1 (2011), 112–118.

[37] Bhekisipho Twala. 2009. An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence* 23, 5 (2009), 373–405.

[38] B Twala, M Cartwright, and M Shepperd. 2005. Comparison of various methods for handling incomplete data in software engineering databases. In *ESEM*. 234–239.

[39] O. Wahltinez et al. 2020. COVID-19 Open-Data: Curating a fine-grained, global-scale data repository for SARS-CoV-2. (2020). https://goo.gle/covid-19-open-data Work in progress.

[40] Jianmin Wang, Shaoxu Song, Xiaochen Zhu, and Xuemin Lin. 2013. Efficient recovery of missing events. *Proceedings of the VLDB Endowment* 6, 10 (2013), 841–852.

[41] Ziheng Wei and Sebastian Link. 2019. Embedded functional dependencies and data-completeness tailored database design. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1458–1470.

[42] Mike Wojnowicz, Ben Cruz, Xuan Zhao, Brian Wallace, Matt Wolff, Jay Luan, and Caleb Crable. 2016. Influence sketching: Finding influential samples in large-scale regressions. In *Big Data*. 3601–3612.

[43] Jing Xia, Shengyu Zhang, Guolong Cai, Li Li, Qing Pan, Jing Yan, and Gangmin Ning. 2017. Adjusted weight voting algorithm for random forests in handling missing values. *Pattern Recognition* 69, 1 (2017), 52–60.

[44] Jianye Yang, Wu Yao, and Wenjie Zhang. 2021. Keyword search on large graphs: A survey. *Data Science and Engineering* 6, 2 (2021), 142–162.

[45] Jinsung Yoon, James Jordon, and Mihaela Schaar. 2018. GAIN: Missing data imputation using generative adversarial nets. In *ICML*. 5675–5684.

[46] Aoqian Zhang, Shaoxu Song, Yu Sun, and Jianmin Wang. 2019. Learning individual models for imputation. In *ICDE*. 160–171.

[47] Yazhong Zhang, Hanbing Zhang, Zhenying He, Yinan Jing, Kai Zhang, and X Sean Wang. 2021. Parrot: A progressive analysis system on large text collections. *Data Science and Engineering* 6, 1 (2021), 1–19.

[48] Zhengyou Zhang, Michael Lyons, Michael Schuster, and Shigeru Akamatsu. 1998. Comparison between geometry-based and gabor-wavelets-based facial expression recognition using multi-layer perceptron. In *FG*. 454–459.