SMITH: A Self-supervised Downstream-Aware Framework for Missing Testing Data Handling

Chih-Chun Yang[1], Cheng-Te Li[2(✉)], and Shou-De Lin[1]

[1] National Taiwan University, Taipei, Taiwan
{r08922050,sdlin}@csie.ntu.edu.tw
[2] National Cheng Kung University, Tainan, Taiwan
chengte@ncku.edu.tw

Abstract. Missing values in testing data has been a notorious problem in machine learning community since it can heavily deteriorate the performance of downstream model learned from complete data without any precaution. To better perform the prediction task with this kind of downstream model, we must impute the missing value first. Therefore, the imputation quality and how to utilize the knowledge provided by the pretrained and fixed downstream model are the keys to address this problem. In this paper, we aim to address this problem and focus on models learned from tabular data. We present a novel Self-supervised downstream-aware framework for MIssing Testing data Handling (SMITH), which consists of a transformer-based imputation model and a downstream label estimation algorithm. The former can be replaced by any existing imputation model of interest with additional performance gain acquired in comparison with that of their original design. By advancing two self-supervised tasks and the knowledge from the prediction of the downstream model to guide the learning of our transformer-based imputation model, our SMITH framework performs favorably against state-of-the-art methods under several benchmarking datasets.

Keywords: Missing testing data · Downstream-aware · Transformer · Self-supervised learning · Tabular data

1 Introduction

Missing values in testing tabular data can heavily deteriorate the performance of downstream model learned from complete data. Despite meticulous control over the data collection pipeline, missing data arise under several circumstances such as the malfunctioning of storage device or the privacy concern that customers are reluctant to provide such information. Regarding this crucial issue raised by the missing values, previous methods addressing the missing testing data problem fall into two categories C1: Take precaution during the learning of downstream

J. Gama et al. (Eds.): PAKDD 2022, LNAI 13281, pp. 499–510, 2022.
https://doi.org/10.1007/978-3-031-05936-0_39

model [10,12]. C2: Impute the missing testing data then feed it into the downstream model as normal inference pipeline[1,4,8,11,12]. Although prevention is better than cure, one major drawback of C1 is that most existing downstream models do not consider the missing issue during designing their architecture, so they usually can not handle missing data. Therefore, C2 is usually preferred since it is applicable to most of existing downstream methods.

To perform imputation on missing testing data, previous work [8] exploits the instance correlation by statistically assuming that similar instances have similar feature. Other methods such as [1,11,12] additionally exploit the correlation among features to predict the missing value by learning a prediction model taking known features as input and estimating the missing value. However, the former suffers from high performance variance considering the wide variety of data distribution of different datasets, and the latter provides imputation of poor quality if the correlation between missing and known features is weak. Aside from the potential issues of imputation quality, previous imputation methods [1,4,8,11] focus on filling missing value via knowledge from the observed feature only. However, they neglect that the information provided by the prediction of pre-trained downstream model could be beneficial. To summarize, missing values in testing data is a challenging problem since the potential issues of the weak correlation among features and the over ideal statistic assumption can lead to poor imputation quality. The imputation methods can only learn from incomplete data, which further increase the difficulty of estimating missing value. Besides, previous methods fail to exploit the beneficial knowledge from the prediction of downstream model.

This paper presents a novel downstream-aware framework, **S**elf-supervised downstream-aware **MI**ssing **T**esting data **H**andling (SMITH), to provide better prediction with missing testing data. SMITH follows the pipeline of C2 to address the problem of missing data during testing phase. We present a transformer-based imputation model that exploits the feature correlation by co-relating input features with self-attention mechanism to provide accurate estimation about missing value. To effectively optimize our imputation model, we design two self-supervised tasks to guide the learning. The first is the masking and prediction task, i.e., we mask the known features then requires our model to predict the masked values. The second is the missing adversarial task, which serves as the regularization term to prevent our model from predicting missing values that is out of regular feature distribution. For leveraging the knowledge from the downstream prediction, we propose a downstream label estimation algorithm, which provides a more confident prediction by aggregating the predictions of similar neighbor instances. The downstream label estimation has two potential usages. One is to be performed on the imputation model optimized already. The other is to serve as extra aid during the learning of imputation model by maximizing the probability of the most confidential label. Besides, our SMITH framework also provides the flexibility of model integration, i.e., it can be applied to any of existing imputation methods of interest by simply replacing the imputation module. We highlight the contribution of this paper below:

- We present a novel downstream-aware framework, SMITH, which performs imputation on the missing testing data through exploiting the knowledge from the downstream model.
- The imputation module in SMITH is realized by a novel transformer-based learning, which is guided by two self-supervised tasks to perform imputation of high quality on the missing testing data.
- The downstream-aware nature of SMITH provides the flexibility of affording existing imputation methods to ameliorate their imputation performance.
- Experiment results show that SMITH outperforms state-of-the-art imputation methods on several benchmarking datasets for downstream predictions.

2 Related Work

Missing Data Imputation. Before the breakthrough of neural network, missing data imputation has already been a well-known research topics in different domain. In this work, we focus on handling imputation on tabular data. For the non-neural-based imputation methods, Mean imputation fill the missing value with global average while KNN imputation [8] fill it with the average of top-k similar instances. However, the performance of such imputation methods heavily relies on the distribution of input data. Other methods typically utilize the correlation among input features by learning a prediction model. For example, MICE [1] adopts multiple chained equations to predict the missing value by learning a regressor for each missing feature with the rest features served as the input for prediction. Nevertheless, the capability of previous non-neural based prediction models is typically not sufficient to give an accurate estimation for the missing value and thus could perform even worse than the non-learning based methods during the circumstances that the correlation among features is weak. With the progress of neural network nowadays, recent neural-based methods directly learn a black-box imputation model by exploiting the strong data fitting capability and robustness of neural network. Additionally, they do not rely on assumptions about the distribution of data. For example, GAIN [11] proposed a GAN-like [5] framework comprised of a generator predicting the missing value and a discriminator measuring the quality of the prediction. By backpropagating the goodness of prediction to the generator, this method outperforms previous traditional learning-based method on several datasets and demonstrate the possibility of bringing neural network into this research topic. However, the optimization of GAIN is more like learning an autoencoder with an extra discriminator, which could possibly result in learning an identity mapping from the input feature to the output. Another neural-based framework, GRAPE [12], adopts graph neural network to address the missing data problem, which constructs a bipartite graph with feature and each observation as nodes and the feature value as the edge. By formulating the imputation as the edge prediction task between feature and observation node, GRAPE demonstrates huge improvement over previous works on several regression task and is claimed to be the state-of-the-art method for missing data imputation.

3 The Proposed SMITH Framework

3.1 Notations and Problem Statement

Now we define the notations. A tabular dataset consists of N instances $\mathcal{D} = \{(x_i, m_i, y_i)\}_{i=1}^{N}$, in which $x_i \in R^d$ that denotes the features vector of each instance, m_i denotes the binary mask with values in $\{0, 1\}^d$ to indicate whether a certain feature is missing, and y_i denotes the ground truth label that is only for evaluation. Besides, we consider the most general missing scenario, missing completely at random *(MCAR)*.

The goal of missing testing data problem is to maximize the performance of the downstream prediction task performed with a fixed classifier $\mathcal{C}$, which is trained on the complete training data while taking missing testing data as input during prediction. To have the fixed downstream classifier to perform inference on missing testing data, an imputation model $\mathcal{M}$ is required to fill the missing value first, taking (x_i, m_i) as input and outputting imputed data $\hat{x}_i$. Thus, the quality of the imputation by $\mathcal{M}$ plays the key role for the downstream performance.

3.2 SMITH Framework

As depicted in Fig. 1, SMITH consists of an imputation model $\mathcal{M}$ and a downstream label estimation algorithm. The former learns imputation from incomplete testing data, and the latter improves the performance further by leveraging knowledge from the downstream prediction task. While the overall framework is optimized, we can perform inference on the missing testing data without exploiting the ground-truth downstream label. With the above-mentioned framework, although testing data contains missing values and the downstream model can only take complete data as input, the downstream task can be achieved by performing imputation on the missing testing data (x_i, m_i), then feeding the imputed features $\hat{x}_i$ into the downstream model to acquire the downstream prediction $\hat{y}_i$. It is worth noting that the imputation model in our framework can be replaced by any existing imputation methods to meet the desire of different downstream tasks. We will present the downstream label estimation in Sect. 3.2, and leave the imputation model introduced in Sect. 4.

3.3 Downstream Label Estimation

Although the ground-truth labels during the testing phase is not available, we propose a downstream label estimation (DLE) algorithm that can exploit favorable knowledge from the downstream prediction to improve the performance. Our downstream label estimation algorithm follows the assumption that instances with similar feature values should be within the same class. For each instance, we perform voting within the prediction of top-k similar instances to give a more confident estimation of downstream label, denoted as $\tilde{y}_i$. Considering the incompleteness of testing data, we perform imputation with the imputation

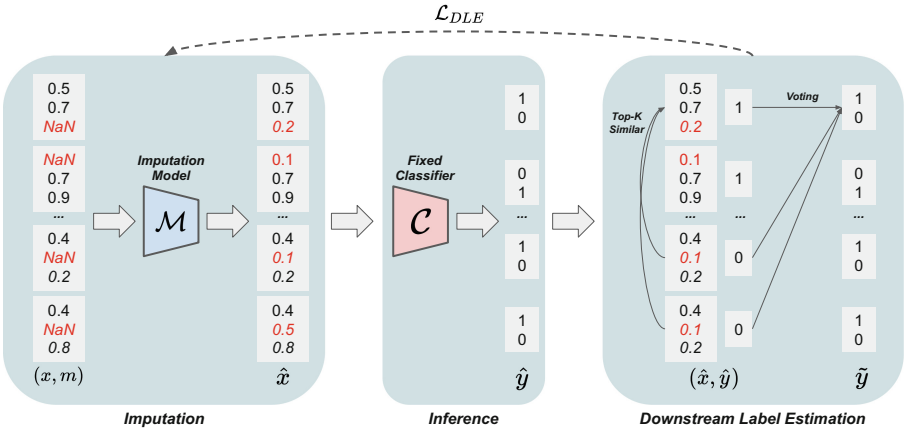

Fig. 1. The proposed SMITH framework.

Algorithm 1: Downstream Label Estimation

Input: Imputation Model $\mathcal{M}$, Downstream Classifier $\mathcal{C}$
Data: Testing data $\mathcal{D} = \{(x_i, m_i)\}_{i=1}^{N}$
Output: Estimated downstream label $\tilde{y}$

Perform imputation on missing data $\mathcal{D}$ with imputation model $\mathcal{M}$ and get imputed data $\hat{x}$.
Get downstream prediction $\hat{y}$ with downstream classifier $\mathcal{C}$.
foreach x_i **do**
    Calculate the similarity between $\hat{x}_i$ and all the other instances $\hat{x}$.
    Find the top-k similar instances according to similarity.
    Perform voting within the prediction of top-k neighbors and acquire the estimated label $\tilde{y}_i$.
end

model $\mathcal{M}$ to acquire data with full feature values, $\hat{x}_i$, then calculate the instance similarity. The full DLE pipeline is detailed in Algorithm 1.

The downstream label estimation can be exploited in two ways, within and after the optimization of imputation model. The former "within imputation" is applicable to neural network-based methods only while the latter "after imputation" is for any supervised learning methods. More specifically, for neural network-based imputation methods like GAIN [11] and GRAPE [12], we can backpropagate the extra knowledge of the estimated label to help the learning of imputation model by maximizing the downstream probability of the corresponding label $\tilde{y}_i$, in which cross-entropy can be used as the objective. We denote this learning objective as $\mathcal{L}_{DLE}$, given by:

$$\mathcal{L}_{DLE} = -\frac{1}{N}\sum_{i=1}^{N} \tilde{y}_i^T \log(p_i) \tag{1}$$

where p_i denotes the vector of label prediction probability estimated by the downstream classifier $\mathcal{C}$, and $\tilde{y}_i$ is the one-hot vector of estimated label. During evaluation, we use $\hat{y}_i$ as the testing prediction result since the knowledge from downstream prediction is already incorporated into the imputation model. As for the non-neural network based method like Mean, KNN [8] and MICE [1], $\tilde{y}_i$ is considered.

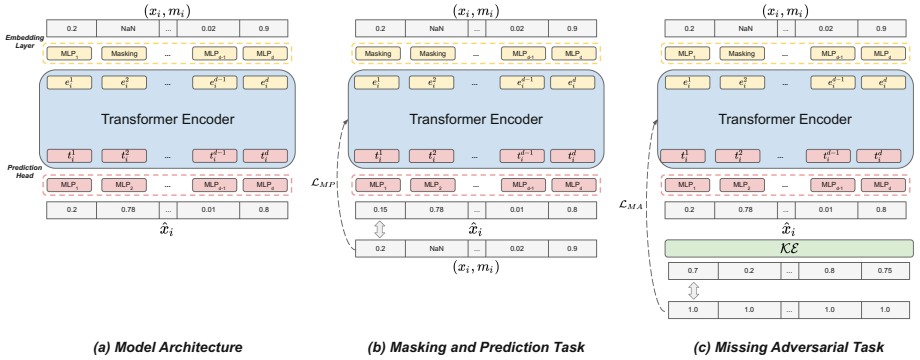

Fig. 2. Proposed transformer-based imputation model and its learning: (a) Model Architecture, (b) Masking and Prediction Task, and (c) Missing Adversarial Task

4 The Proposed Transformer-based Imputation Model

In this section, we introduce the learning of our transformer-based imputation model. The goal of the imputation model is to estimate the missing value based on the known input feature. As shown in Fig. 2(a), our imputation model consists of three modules, including the feature-specific embedding layer, transformer encoder, and feature-specific prediction head. The forward pass of our model is detailed as follows. First, we encode each known feature x_i^j in instance x_i into feature embedding $e_i^j \in R^k$, with feature-specific embedding layer respectively. For the numerical feature, we apply a two-layer MLP for each feature, which transforms the input scalar into a one-dimensional embedding. For the categorical feature, we simply use an embedding layer, similar to that of the word embedding [7]. As for the missing feature values, we adopt a special embedding as the feature embedding for each missing feature to indicate the missingness. After the above-mentioned encoding pipelines are completed, an extra embedding for each feature is added to emphasize the feature modality information, similar to the positional encoding [3]. As a result, we transform each instance from one-dimensional feature vector x_i into two-dimensional one, $e_i \in R^{d\times k}$. As the encoding process is completed, we feed the feature-specific representation e_i into the transformer encoder that leverages the correlation of each feature via self-attention [9] and output the representation t_i. The final prediction for imputation

is acquired by feeding the representation t_i into the feature-specific prediction head, transforming the two-dimensional representation into a one-dimensional one, the same shape as the input feature. To optimize our imputation model, we introduce two self-supervised tasks, masking and prediction task and missing adversarial task, to guide its learning. The former enforces our model to estimate the missing values with the known features, and the latter prevents our model from predicting values which is out of normal data distribution.

4.1 Masking and Prediction Task

In this task, we aim at exploiting the correlation among the known feature values to guide the learning of our imputation model. Aside from the missing feature values which are unobserved at the testing stage, we perform extra masking on known feature values selected randomly, and require our model to recover them. Unlike the learning of autoencoder that the ground-truth feature value is already in its input, our masking operation reinforces the awareness between missing and known features, and prevents our model from identically mapping the input to the output. We illustrate this task in Fig. 2(b). For numerical features, we use mean square error as the learning objective. As for categorical feature, cross-entropy is considered. We denote this learning objective as $\mathcal{L}_{MP}$ and formally define it as follows:

$$\mathcal{L}_{MP} = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} ||\hat{x}_i^j - x_i^j||_2, & \text{if } x^j \text{ is numerical.} \\ -\frac{1}{N}\sum_{i=1}^{N} (x_i^j)^T \log(\hat{x}_i^j), & \text{if } x^j \text{ is categorical.} \end{cases} \tag{2}$$

4.2 Missing Adversarial Task

In addition to the concrete guidance provided by the known feature values, in this task, we prevent our model from predicting feature values that is out of their reasonable distribution with the help of an extra module, known estimator $\mathcal{KE}$. The $\mathcal{KE}$ estimates the goodness of the prediction by learning to distinguish known features from the predicted features generated from the imputation model. As depicted in Fig. 2(c), the known estimator $\mathcal{KE}$ takes the prediction from the imputation model $\mathcal{M}$, and outputs scores ranging from 0 to 1 to indicate the goodness of each feature value. The learning of $\mathcal{KE}$ is formulated as a feature-wise binary classification task. Formally, the learning objective for $\mathcal{KE}$ is as follows,

$$\mathcal{L}_{Est} = -\frac{1}{N}\sum_{i=1}^{N} m_i^T \log(\mathcal{KE}(\hat{x_i})) \tag{3}$$

To backpropagate the knowledge of the prediction quality to the imputation model, we fix the model parameters of the known estimator $\mathcal{KE}$, and update those of the imputation model $\mathcal{M}$ to maximize the goodness score. We denote this learning objective as $\mathcal{L}_{MA}$, given by:

$$\mathcal{L}_{MA} = -\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}^T \log(\mathcal{KE}(\hat{x_i})). \tag{4}$$

Algorithm 2: Full Optimization Pipeline

Input: Imputation Model $\mathcal{M}$, Downstream Classifier $\mathcal{C}$, Known Estimator $\mathcal{KE}$

Data: Testing data $\mathcal{D} = \{(x_i, m_i)\}_{i=1}^{N}$ with missing values

```
foreach epoch do
    Get imputation value with imputer M.
    Get known estimation with KE.
    Calculate known estimation loss L_Est.
    Update known estimator KE.
    Calculate mask prediction loss L_MP.
    Calculate missing adversarial loss L_MA.
    if epoch ≠ 0 then
        Calculate downstream label estimation loss L_DLE.
        Update imputer M.
        Update estimated downstream label ŷ_i following Algorithm 1.
    else
        Update imputer M.
        Get estimated downstream label ŷ_i following Algorithm 1.
    end
end
```

Following the general learning pipeline of Generative Adversarial Network (GAN) [5], we train the known estimator and our imputation model iteratively with the objectives mentioned above.

4.3 The Overall Learning Objective

Now we summarize all the learning objectives introduced in the sections above. For the learning of our imputation model, we have $\mathcal{L}_{SST}$ sum up the two self-supervised objectives related to imputation as follows: $\mathcal{L}_{SST} = \mathcal{L}_{MP} + \lambda_a \mathcal{L}_{MA}$. After introducing the downstream label estimation into our imputation model, the full learning objective is formulated as below: $\mathcal{L} = \mathcal{L}_{SST} + \lambda_d \mathcal{L}_{DLE}$, where λ_a and λ_d denote the balancing factors of the missing adversarial task and the utilization of knowledge from downstream prediction during optimization. Since the guidance provided by these objectives is imprecise, in practice, we set relatively small learning weights, e.g., 0.01, to them. Note that all the existing neural network-based imputation methods can be incorporated into our framework by replacing the imputation model and the $\mathcal{L}_{SST}$ will be substituted with their learning objective accordingly. We detail the full optimization pipeline of the integration of our transformer-based imputation model into our SMITH framework in Algorithm 2.

5 Experiments

In this section, we conduct extensive experiments to quantitatively verify the effectiveness of SMITH on the task of missing testing data.

Table 1. Statistics of datasets.

Statistics	Breast	Spam	Electric	Letter	Plate	Bean	Wifi	Wine	Digit	Yeast
Abbr.	BR	SP	EL	LE	PL	BE	WIF	WIN	DI	YE
Instance	540	4601	10000	20000	1941	13611	2000	4898	10992	1484
Attribute	18	57	12	16	27	16	7	11	16	8
Class	2	2	2	26	7	7	4	7	10	10

5.1 Experimental Setup

We conduct experiments on 10 real world datasets from the UCI Machine Learning Repository [2], and the statistics is listed in Table 1. Since these datasets are originally complete, we drop features following the missing completely at random *(MCAR)* setting to simulate the missing testing data problem. In all experiments, we split the dataset into training/validation/testing set with 60/10/30% ratio. The downstream classifier is optimized on training set with complete instances and corresponding labels. The imputation model is optimized on the testing set with instances containing missing features only and validated with the validation set which is similar to the former but with the ground-truth label to select the best checkpoint. Following the same pipeline as previous works [11,12], we scale feature values to [0-1] with MinMax Scalar [6] as the feature preprocessing. We use a 3-layer MLP with *tanh* as the downstream classifier. For the analyses, we conduct experiments with the setting mentioned above 5 times with different random seeds and report the mean accuracy.

We compare SMITH with the following imputation methods: (a) **Mean**: Impute the missing value with the mean of observed feature value; (b) **KNN** [8]: Impute with the mean value among top-k most similar instances; (c) **MICE** [1]: Impute the missing value based on a simple prediction model conditioned on the other feature and optimized on instances with observed value; (d) **GAIN** [11]: State-of-the-art neural method with generative adversarial training; (e) **GRAPE** [12]: State-of-the-art neural method following the graph architecture.

5.2 Experimental Results

Comparison of Imputation Models. In this section, we compare the imputation capability of SMITH with competing methods on different levels of missing rate. To focus on the imputation capability, knowledge from the downstream model is not involved in these experiments. We found that though some neural-based methods are claimed to be the state-of-the-art, they could lose to non-neural based imputation methods on several datasets. Since there is no single

Table 2. Performance comparison of imputation models on different missing rates.

Method	BR	SP	EL	LE	BE	WIF	DI	PL	WIN	YE	Rank
30% Missing Rate											
Mean	94.5	87.96	75.4	45.19	76.16	85.17	70.02	57.39	46.45	43.73	5.5
KNN	94.69	88.16	74.75	60.45	90.01	86.47	91.06	65.57	46.99	43.06	4.2
MICE	**96.96**	88.22	76.79	52.46	90.69	91.87	87.29	65.4	46.88	44.0	3.1
GAIN	94.97	88.83	75.25	51.44	88.9	88.43	83.99	62.92	46.49	42.43	4.6
GRAPE	95.67	**89.45**	76.46	64.46	90.81	93.83	94.9	66.49	47.09	43.51	2.3
SMITH	96.26	88.72	**76.99**	**68.71**	**90.84**	**94.1**	**95.29**	**66.7**	**47.16**	**44.72**	**1.3**
50% Missing Rate											
Mean	92.28	81.93	71.36	28.77	55.78	69.37	50.07	44.95	46.26	40.76	5
KNN	93.68	82.06	68.23	32.59	76.62	77.3	59.39	57.94	46.05	39.69	4.85
MICE	**94.85**	84.23	71.59	36.77	87.25	80.6	69.77	58.59	46.14	40.4	3
GAIN	93.62	84.59	71.33	33.79	83.73	76.57	66.7	56.46	46.0	40.49	4.3
GRAPE	93.68	**86.09**	72.37	47.42	**89.47**	84.57	**87.8**	62.13	46.11	40.27	2.35
SMITH	94.27	84.33	**72.66**	**48.18**	89.35	**84.73**	86.77	**62.3**	**46.34**	**41.26**	**1.5**
70% Missing Rate											
Mean	88.77	73.19	67.85	16.41	35.57	50.8	30.48	38.01	45.78	36.31	4.85
KNN	88.07	73.74	63.95	18.08	63.55	61.83	41.87	48.73	45.78	34.7	4.45
MICE	**92.05**	77.68	68.03	19.4	69.2	63.93	39.16	47.73	**46.03**	35.19	2.9
GAIN	87.13	77.39	67.6	14.96	59.49	56.63	40.99	47.66	45.63	35.51	4.7
GRAPE	90.29	**80.62**	67.23	**25.9**	**85.47**	67.73	**65.3**	**54.47**	45.89	35.42	2.2
SMITH	91.46	74.78	**68.25**	24.6	85.2	**68.0**	62.57	52.78	45.97	**36.63**	**1.9**

method outperforming the others among all datasets, we compare them by the average ranking. Table 2 shows SMITH outperforms all competitors. However, as the missing rate increases, the ranking of our model gradually decreases.

Analysis on Downstream Label Estimation Algorithm. We investigate whether our downstream label estimation (DLE) algorithm can improve the performance of downstream task. Besides, we compare the two strategies of incorporating the algorithm into existing imputation methods, within or after the optimization of imputation, denoted as *single* (-s) and *iterative* (-i) respectively. The results are reported in Table 3. In comparison with the results in Table 2, we see that our algorithm improves the performance of all imputation methods. As for the comparison of different strategies, we observe that the option *iterative* always outperforms the option *single* on 30% and 50% missing rates. This is expected since the extra knowledge of downstream model is backpropagated to the learning of imputation. However, in the setting of 70% missing rate, the performance of *iterative* is worse than *single* in some datasets. We attribute this phenomenon to the uncertainty of the imputation learning on 70% missing rate. Incorporating such an objective could bring additional noise since the learning is already difficult. Consequently, our model still outperforms all competitors across all missing rates after we advance the downstream label estimation.

Ablation Study. To verify the effectiveness of each component in SMITH, we conduct ablation analyses on two self-supervised tasks and the downstream label estimation (DLE) algorithm on 10 datasets with 30% missing values. Table 4 shows the effectiveness of each learning objective is confirmed.

Table 3. Performance comparison between *iterative* and *single* learning strategies.

Method	BR	SP	EL	LE	BE	WIF	DI	PL	WIN	YE	Rank
30% Missing Rate											
Mean-s	95.02	88.01	75.62	51.42	81.13	87.97	76.51	58.16	46.55	43.92	5.4
KNN-s	94.91	88.23	74.98	64.31	90.45	88.79	91.37	65.82	47.03	43.23	4.5
MICE-s	97.01	88.31	77.01	54.57	90.81	92.76	88.81	65.77	46.91	44.03	3.5
GAIN-s	95.11	89.01	75.34	62.48	89.98	93.1	92.46	63.39	46.52	43.73	4.2
GAIN-i	95.32	89.11	75.43	63.12	90.03	93.99	93.67	63.57	46.6	44.05	3.9
GRAPE-s	95.92	89.52	76.52	65.83	90.96	93.87	95.16	66.57	47.12	44.02	2.2
GRAPE-i	96.07	**89.67**	76.66	67.59	91.02	94.47	95.34	66.71	47.19	44.22	2.1
SMITH-s	97.03	88.91	77.08	69.23	91.08	94.33	95.37	66.85	47.29	44.91	1.2
SMITH-i	**97.13**	88.98	**77.23**	**70.12**	**91.17**	**94.61**	**95.48**	**66.93**	**47.38**	**44.97**	**1.2**
50% Missing Rate											
Mean-s	93.15	82.07	71.45	31.88	70.66	73.48	61.73	48.12	46.47	41.23	5
KNN-s	94.03	82.15	69.76	39.92	80.89	80.21	65.88	58.23	46.33	40.11	4.8
MICE-s	95.12	84.45	71.88	41.98	88.56	83.11	72.19	59.44	46.41	40.72	3.2
GAIN-s	93.78	84.65	71.42	41.1	87.72	83.13	80.91	57.51	46.07	40.77	4
GAIN-i	93.98	84.68	71.48	41.59	88.62	83.22	81.27	57.98	46.31	41.03	3.7
GRAPE-s	93.89	86.23	72.44	50.46	89.51	85.07	88.64	62.31	46.19	40.58	2.5
GRAPE-i	94.04	**86.31**	72.53	50.77	**89.63**	85.18	**88.89**	62.69	46.2	40.95	2.4
SMITH-s	94.98	84.47	72.71	51.24	89.47	85.34	88.43	62.42	46.51	41.31	1.5
SMITH-i	**95.16**	84.51	**72.83**	**51.45**	89.52	**85.45**	88.63	**62.78**	**46.62**	**41.39**	**1.4**
70% Missing Rate											
Mean-s	89.11	73.22	68.05	18.12	50.23	53.41	40.77	43.38	45.8	36.57	5
KNN-s	88.93	73.78	65.59	19.2	67.88	62.78	45.91	49.22	45.91	35.11	4.7
MICE-s	92.66	77.72	68.13	21.01	71.37	65.77	42.83	48.55	**46.22**	35.56	2.9
GAIN-s	91.46	77.63	67.79	17.12	76.02	59.37	52.71	48.18	45.73	35.57	4.1
GAIN-i	88.77	76.78	67.19	16.01	74.94	56.83	47.21	48.32	45.54	35.73	4.5
GRAPE-s	91.35	**80.96**	67.44	**26.45**	85.63	68.6	**68.7**	**54.74**	45.96	35.51	2.4
GRAPE-i	90.76	79.32	67.63	26.15	**85.82**	68.63	65.65	54.37	45.97	35.06	2.3
SMITH-s	92.21	76.15	68.31	25.78	85.47	68.83	67.16	53.26	46.09	**36.81**	1.9
SMITH-i	**92.98**	76.57	**68.39**	25.81	85.67	**68.89**	65.51	53.78	46.13	36.74	**1.8**

Table 4. Results of ablation study.

Setting	BR	SP	EL	LE	BE	WIF	DI	PL	WIN	YE
$\mathcal{L}_{MP}$	95.74	88.01	76.51	68.23	90.75	93.88	95.01	66.31	46.89	44.39
$+\mathcal{L}_{MA}$	96.26	88.72	76.99	68.71	90.84	94.1	95.29	66.7	47.16	44.72
$+\mathcal{L}_{DLE}$	**97.13**	**88.98**	**77.23**	**70.12**	**91.17**	**94.61**	**95.48**	**66.93**	**47.38**	**44.97**

6 Conclusion

In this paper, we propose a novel downstream-aware framework, SMITH, comprised of a transformer-based imputation model learned from two self-supervised tasks and a downstream label estimation algorithm to handle missing data during prediction. SMITH is flexible to be applied to any existing imputation methods. By advancing the extra knowledge from the downstream model, we demonstrate improvement over 10 benchmark datasets. With SMITH, we outperform previous state-of-the-art methods regarding the overall average ranking. Extensive experiments are conducted to quantitatively verify the effectiveness of our method.

Acknowledgements. This material is based upon work supported by Taiwan Ministry of Science and Technology (MOST) under grant numbers 110-2634-F-002-050, 110-2221-E-006-136-MY3, 110-2221-E-006-001, and 110-2634-F-002-051.

References

1. Buuren, S., Groothuis-Oudshoorn, C.: Mice: Multivariate imputation by chained equations in R. J. Stat. Softw. **45**, 1–67 (2011)
2. Dua, D., Graff, C.: UCI machine learning repository (2017)
3. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70. pp. 1243–1252. ICML 2017 (2017)
4. Gondara, L., Wang, K.: Mida: multiple imputation using denoising autoencoders. In: PAKDD (2018)
5. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems. vol. 27 (2014)
6. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets. Cambridge University Press, 3 edn. Cambridge 2020)
7. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: 1st International Conference on Learning Representations, ICLR 2013, Workshop Track Proceedings (2013)
8. Troyanskaya, O., et al.: Missing value estimation methods for DNA microarrays. Bioinformatics **17**(6), 520–525 (2001)
9. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
10. Yi, J., Lee, J., Kim, K.J., Hwang, S.J., Yang, E.: Why not to use zero imputation? correcting sparsity bias in training neural networks. In: International Conference on Learning Representations (2020)
11. Yoon, J., Jordon, J., van der Schaar, M.: GAIN: Missing data imputation using generative adversarial nets. In: Proceedings of the 35th International Conference on Machine Learning. pp. 5689–5698 (2018)
12. You, J., Ma, X., Ding, Y., Kochenderfer, M.J., Leskovec, J.: Handling missing data with graph representation learning. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. pp. 19075–19087 (2020)