공학석사 학위논문

테이블 데이터 품질 개선을 위한 셀프 어텐션 기반 보간 기법

A self-attention-based imputation method for improving tabular data quality

2022년 2월

서울시립대학교 대학원

전자전기컴퓨터공학과

이 도 훈

테이블 데이터 품질 개선을 위한 셀프 어텐션 기반 보간 기법

A self-attention-based imputation method for improving tabular data quality

지도교수 김 한 준

이 논문을 공학석사학위 논문으로 제출함

2021년 12월

서울시립대학교 대학원

전자전기컴퓨터공학과

이 도 훈

이 도 훈의 공학석사 학위논문을 인준함.

심사위원장 최 성 종 인

심 사 위 원 김 한 준 인

심 사 위 원 송 경 우 인

2021년 12월

서울시립대학교 대학원

Abstract

Recently, data-driven decision-making has attracted great interest; this requires high-quality datasets. However, real-world datasets often feature missing values (for various reasons), rendering data-driven decision-making inaccurate. Many studies have sought to derive missing value imputation methods. Many conventional machine-learning-based imputation methods that handle tabular data can deal with only numerical columns, or are time-consuming and cumbersome because they create an individualized predictive model for each column. Therefore, we have developed a novel imputational neural network that we term the 'Denoising Self-Attention Network (DSAN)'. This network can deal with tabular datasets containing both numerical and categorical variables. The DSAN learns robust feature expression vectors by combining self-attention and denoising techniques, and can predict multiple, appropriate substituted values in parallel (via multi-task learning). To verify the validity of the method, we performed data imputation experiments after arbitrarily generating missing values for several real-world tabular datasets. We evaluated both imputational and downstream task performances.

Keywords Attention Network, Data Quality, Deep Learning, Machine-learning, Missing Value Imputation, Multi-Task Learning

공학석사 학위논문

테이블 데이터 품질 개선을 위한 셀프 어텐션 기반 보간 기법

A self-attention-based imputation method for improving tabular data quality

2022년 2월

서울시립대학교 대학원

전자전기컴퓨터공학과

이 도 훈

Contents

List of Tables

List of Figures

Chapter 1. Introduction

As the amount of data increases, data-driven decision-making via machine-learning has become increasingly important in many fields. The results are greatly affected by data quality; good-quality data are essential. The most common problem is missing values, which damage the data pipelines and render the results of data-driven tasks inaccurate. If a machine-learning model is trained using incomplete datasets, the inferred results may be biased. [4] For example, many missing values arise because customers do not wish to give personal information when asked to comment on e-commerce or mobile advertising domains. In such cases, data scientists or machine-learning engineers must clean the datasets to ensure meaningful results. Missing value imputation (MVI) is the most commonly adopted solution. MVI seeks to replace the missing values with substituted values estimated based on the observed values. [17]

In general, MVI methods use a statistical or machine-learning technique to replace missing values with substituted values. Statistical MVI methods employ descriptive statistics such as the mean/mode; they are thus simple and fast, but inaccurate. Machine-learning-based MVI methods have recently been studied to ensure more accurate results. Such methods generate or predict substituted values using trained generative or predictive models. MVI methods employing generative models include Generative Adversarial Imputation Nets (GAIN) [25] and denoising autoencoder (DAE; for multiple imputations) [10]; MVI methods that feature predictive models include MissForest [21] and Datawig. [5] Genera-

tive model-based MVI methods cannot generate substituted values for categorical variables; such models cannot handle tabular data that contain both numerical and categorical parameters. Meanwhile, predictive model-based MVI methods train models for each variable, and can thus deal with tabular data. However, it is both cumbersome and inefficient to train several predictive models.

Here, we have developed a novel, end-to-end, imputational neural network termed the Denoising Self-Attention Network (DSAN), which can handle tabular data. The DSAN combines self-attention-based feature representation and denoising to learn robust features prior to analysis of an incomplete mixed-type dataset. The DSAN is trained (using a multi-task learning [MTL] method) to predict substituted values that are appropriate for the type of variable under consideration. DSAN training proceeds in a self-supervised manner; each received input is re-predicted. During such training, the DSAN learns the interactions among variables and among observed and missing values. Thus, the DSAN predicts substituted values appropriate for each type of variable, and missing values.

To verify the method, we performed experiments evaluating two types of performance: imputation performance and downstream task performance. Imputation performance is the accuracy by which a method imputes a substituted value, and is calculated as the errors between the imputed and original values. Downstream task performance explores whether the imputed dataset can be used to solve other tasks. To explore downstream task performance, a binary classifier was trained using an imputed dataset, and performance was then evaluated.

Chapter 2. Background

Recently, deep neural network-based imputational methods for tabular data have received much attention. Deep neural networks afford several advantages, including end-to-end learning and MTL, possible fusion of multiple modalities (e.g., images and text), and representation learning. Efforts have been made to apply such models to MVI tasks. The representational learning of a deep neural network is used to extract features of tabular data that aid predictive imputation. [5, 14] MTL can simultaneously handle regression and classification tasks; thus, an imputational model can combine imputation task and classification task. [16]

The attention mechanism employed in machine translation [3] has been successfully ported to natural language processing tasks [6, 7] and computer vision.[8] Attempts have been made to incorporate the mechanism into deep neural networks that handle tabular data. TabNet[1] used an attention mechanism to imitate a decision tree that works well when used to handle structured data. TabNet enabled interpretable and efficient learning; sequential attention was used to select the important features of each decision-making stage. TabTransformer[13] used the encoder of a transformer[22] and on a self-attention mechanism to map categorical features into a contextual embedment. TabTransformer outperformed a multi-layer perceptron (MLP) and a tree-based ensemble model, and was very robust against noisy and missing data.

Our imputation method has a special deep learning architecture that is quite different from the existing methods mentioned above. (1) Our method has a self-

attention neural network architecture to realize effective representation learning for noisy data. (2) We use MTL to predict missing values for various column types in imputation task, not to combine imputation task and downstream task. (3) Generally, the attention-based neural networks for tabular data [1, 13] have been proposed to solve the classification problem. In contrast, our method includes the attention mechanism to solve the missing value imputation problem. (4) Our method considers discretized numerical values as categorical values for embedding and self-attention layers.

Chapter 3. Problem Formulation

We seek to transform an incomplete tabular dataset(Fig. 3.1) into a complete dataset(Fig. 3.2) by predicting substitute values that impute missing values. The tabular data of interest include both numerical and categorical variables; we solve regression and classification problems simultaneously.

Let $\boldsymbol{x} = \{\boldsymbol{x}^{num}, \boldsymbol{x}^{cat}\}$ denote an input vector(a data record). An input vector $\boldsymbol{x}$ contains n numerical variables $\boldsymbol{x}^{num} \in \mathbb{R}^n$ and k categorical variables $\boldsymbol{x}^{cat} = \{x_1^{cat}, x_2^{cat}, \cdots, x_k^{cat}\}$. The i-th categorical variable x_i^{cat} has a domain set C_i in which all observed values of the i-th categorical variable x_i^{cat} lie. We use an indicator vector $\boldsymbol{m} = \{\boldsymbol{m}^{num}, \boldsymbol{m}^{cat}\} \in \{0, 1\}^{(n+k)}$ to indicate whether a given variable is missing or not. The indicator value $m_i = 1$ if the corresponding i-th variable(numerical or categorical) is observed; otherwise, $m_i = 0$. These values are later used to compute the loss function (employing only observed values).

Given an input vector $\boldsymbol{x}$, the imputational model predicts $\hat{\boldsymbol{x}}$ using the input values. We then fill the missing values with predicted values. As mentioned above, if a variable is numerical, we reconstruct an observed value; if a variable is categorical, we classify it by targeting the observed value.

age	job	martial	education	default	···	duration	campaign	pdays	poutcome	deposit
59	admin.	married	secondary	no	···	1042	1	**NaN**	0	yes
56	admin.	married	secondary	no	···	1467	1	-1	0	yes
41	technician	married	secondary	no	···	**NaN**	1	-1	0	yes
55	services	married	secondary	no	···	579	1	-1	0	yes
54	admin.	**NaN**	**NaN**	no	···	673	2	-1	0	yes

Figure 3.1: Example of an input dataset with incomplete tabular data. This is the Bank dataset used for performance evaluation in Chapter 5.

age	job	martial	education	default	···	duration	campaign	pdays	poutcome	deposit
59	admin.	married	secondary	no	···	1042	1	**-1**	0	yes
56	admin.	married	secondary	no	···	1467	1	-1	0	yes
41	technician	married	secondary	no	···	**1389**	1	-1	0	yes
55	services	married	secondary	no	···	579	1	-1	0	yes
54	admin.	**married**	**tertiary**	no	···	673	2	-1	0	yes

Figure 3.2: Example of the output data (imputed data) that are sought.

Chapter 4. Proposed Method

Our model features end-to-end MVI of tabular data, as shown in Fig. 4.1. We term the model the Denoising Self-Attention Network (DSAN). There are three modules: A feature representation module, a shared-layer module, and a task-specific module. The first module column-embeds each variable [13] and learns contextual embedding employing the self-attention layers. The model then is trained (via MTL) to predict appropriate substituted values for the different variables. The DSAN features hard parameter sharing: [20] The shared-layer module learns the shared parameters by sharing the hidden layers among all tasks. The task-specific layer module learns the various parameters that should be retained in the (several) task-specific output layers. The DSAN is trained in a self-supervised manner; observed values are reconstructed in a manner similar to that of AutoEncoder. [11]

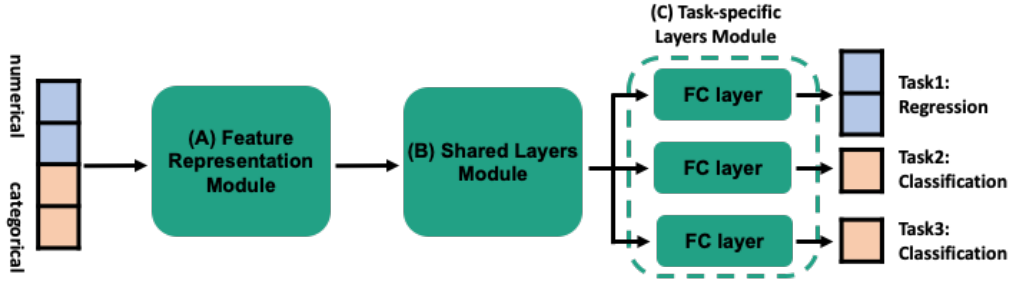

Figure 4.1: An overview of the proposed model.

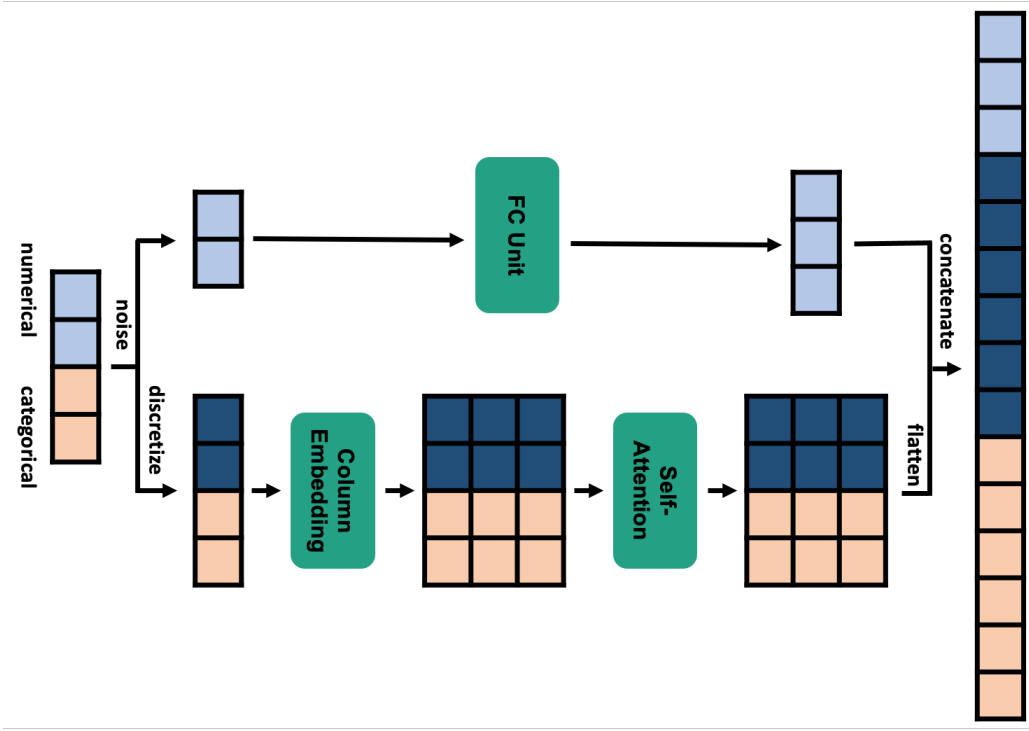

Figure 4.2: An overview of the feature representation module.

4.1 The feature representation module

The feature representation module learns how to extract useful information from incomplete mixed-type inputs, as shown as Fig 4.2. In this section, we explain the details.

4.1.1 Preprocessing

To handle incomplete mixed-type data, we preprocess the input vectors as shown in Fig. 4.3. First, we initialize a missing value(e.g., NaN) to a processible value. A missing numerical variable is initialized to 0; a missing categorical

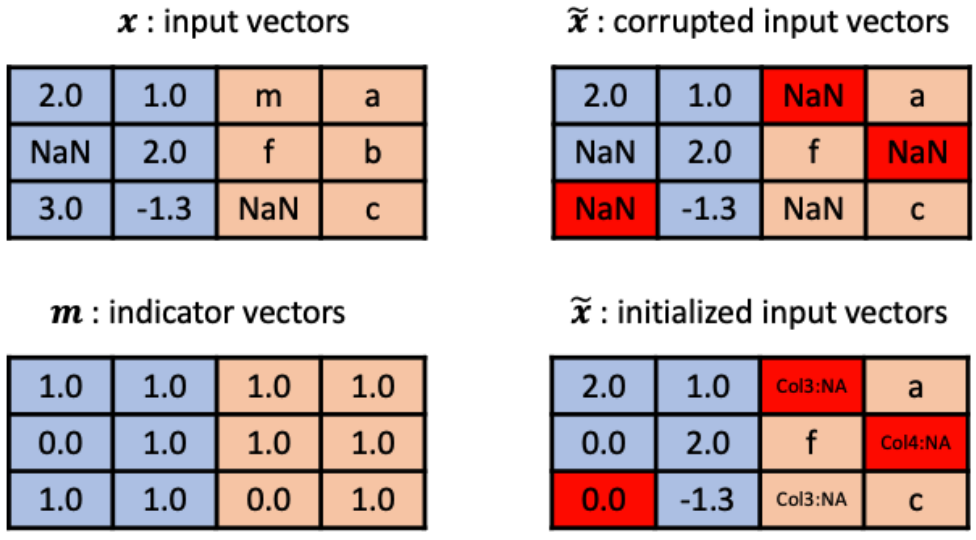

Figure 4.3: An example of model inputs.

variable is initialized to a unique string indicating that the value is missing. For example, if the value for the third column (a categorical variable) is missing, it is initialized to 'Col3:NULL'. Then, we use a denoising technique [23] to learn robust feature representation parameters for incomplete inputs. To this end, a certain percentage α of the observed values are randomly selected and dropped out. The dropped-out values are initialized in the manner described above, but the values of all indicator vectors $\boldsymbol{m}$ are held at 1. We thus obtain corrupted input vectors $\tilde{\boldsymbol{x}}$ that are input into the model.

4.1.2 Column embedding

We embed all inputs into a parametric embedment of dimension d using the embedding layer. Here, the numerical variables $\tilde{\boldsymbol{x}}^{num}$ are discretized (to reduce

computation and ensure regularization). Given an input vector $\{\tilde{\boldsymbol{x}}^{num}, \tilde{\boldsymbol{x}}^{cat}\}$, an embedding matrix $E \in \mathbb{R}^{(n+k)\times d}$ is computed using:

$$E = Embedding(Discretize(\tilde{\boldsymbol{x}}^{num}), \tilde{\boldsymbol{x}}^{cat}) \tag{4.1}$$

where the *Embedding* maps each value to an d-dimensional vector. *Discretize* converts continuous to discrete values using certain rules in a heuristic manner. Our rules are:

$$Discretize(x) = \text{int}(\log_2(x)^2) \tag{4.2}$$

At this time, the DSAN also learns an embedding vector with a unique value that indicates that a value is missing (e.g., 'Col3:NULL'). The information loss caused by missing values is supplemented, and feature representations for the various missing patterns (arbitrarily generated via denoising) are learned. This improves performance; the technique is similar to data augmentation. [24] The embedding matrix E is input to the self-attention layer.

Some information loss is inevitable when discretizing numerical variables. To compensate for this, numerical values that are not discretized are learned in parallel via a fully connected (FC) unit consisting of an FC layer, Layer Normalization [2], the ReLU function, and Dropout. This unit is also active in the subsequent shared-layer module. In the feature representation module, the FC unit employs d nodes as the embedding dimension (to unify feature size). This maintains information on the raw numerical variables, and serves as an input to the shared-layer module, along with the contextual embedding matrix computed via self-attention.

4.1.3 The self-attention layer

As MVI predicts substituted values based on observed values, it is important to learn the relationships between the variables. Therefore, we trained the DSAN to learn the contextual embedments (including the associations) between the variables. We used self-attention to this end; this combines queries, keys, and values into single inputs and engages in feature representation by computing the relationships between elements of the input. We employed multi-head attention; [22] this linearly projects queries, keys, and values h times but with different parametric weights, and (in parallel) runs the attention function h times. It is thus possible to simultaneously focus on information from subspaces featuring different representations. Given an embedding matrix E, the contextual embedding matrix $H \in \mathbb{R}^{(n+k)\times d}$ is computed as follows:

$$H = Concat(\text{head}_1, \cdots, \text{head}_h)W^O \tag{4.3}$$

where $\text{head}_i \in \mathbb{R}^{(n+k)\times(d/h)}$ is the partial attention result, thus that computed by:

$$\text{head}_i = Attention(EW_i^Q, EW_i^K, EW_i^V). \tag{4.4}$$

where the projections are parametric matrices $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d\times(d/h)}$ and $W^O \in \mathbb{R}^{d\times d}$. *Attention* is the scaled, dot-product attention function:

$$Attention(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d/h}})V \tag{4.5}$$

The computed, contextual embedding matrix is flattened into a vector and concatenated with the output of the FC unit. Thus, we obtain a concatenated vector $\boldsymbol{z}_{feature} \in \mathbb{R}^{(n+k+1)\cdot d}$ that is input to the shared-layer module:

$$\boldsymbol{z}_{feature} = Concat(Flatten(H), FCunit(\tilde{\boldsymbol{x}}^{num})) \tag{4.6}$$

4.2 The shared-layer and task-specific layer modules

As mentioned above, the DSAN performs regression and classification tasks in parallel using MTL. Each task is a prediction appropriate for the type of variable. If the variable is numerical, the task is a regression; if the variable is categorical, the task is a classification. We used hard parameter sharing (the most common form of MTL); this reduces overfitting by sharing the hidden layers. The components output by the feature representation module are divided between the shared-layer and the task-specific layer modules. We created the shared-layer module f_{share} by stacking FC units in layers. All units featured the same number of nodes as the dimensions of the concatenated vectors $\boldsymbol{z}_{feature}$'s dimension $(n+k+1)\cdot d$. We used the residual connection approach [12] to enable deep layer learning. Given $\boldsymbol{z}_{feature}$, we computed the output of the shared-layer module $\boldsymbol{z}_{share} \in \mathbb{R}^{(n+k+1)\cdot d}$ using the inputs of the several task layers:

$$\boldsymbol{z}_{share} = f_{share}(\boldsymbol{z}_{feature}) \tag{4.7}$$

The task-specific layer module featured as many FC layers as the number of tasks. During regression, the DSAN also predicts numerical variables. During classification, the DSAN predicts categorical variables separately. Thus, employing $1+k$ FC layers $\{f_{num}, f_1, \cdots, f_k\}$, the DSAN is trained in a self-supervised manner; the observed input is re-predicted. The regression task FC layer f_{num} reconstructs the input numerical values:

$$\hat{\boldsymbol{x}}^{num} = f_{num}(\boldsymbol{z}_{share}) \tag{4.8}$$

and the other k classification task FC layers f_i classify the input categorical values:

$$\hat{\boldsymbol{y}}_i = \sigma(f_i(\boldsymbol{z}_{share})) \tag{4.9}$$

where $\hat{\boldsymbol{y}}_i$ is the probability vector that the target is the i-th categorical value x_i^{cat}. If the size of the domain set of x_i^{cat} is $|C_i| = 2$, this is a binary classification task, and the output $\hat{\boldsymbol{y}}_i$ is thus a scalar. The σ activation function maps this to a probability. If a task is a binary classification, σ is a sigmoid function; if a task features multi-class classification, σ is a softmax function. During MTL, the total loss $\mathcal{L}_{tot}$ is the sum of the losses of each task:

$$\mathcal{L}_{tot} = \mathcal{L}_{num} + \sum_{i=1}^{k} \mathcal{L}_i \tag{4.10}$$

$\mathcal{L}_{num}$ is the regression task loss; the mean squared errors are computed using only the observed values:

$$\mathcal{L}_{num} = \frac{1}{n} \sum_{i=1}^{n} m_i^{num} (x_i^{num} - \hat{x}_i^{num})^2 \tag{4.11}$$

$\mathcal{L}_i$ is the i-th classification task loss; the cross-entropy loss is computed using only the observed values:

$$\mathcal{L}_i = -m_i^{cat} \sum_{j=1}^{|C_i|} y_{i,j} \log(\hat{y}_{i,j}) \tag{4.12}$$

We minimized the total loss $\mathcal{L}_{tot}$; all DSAN parameters are learnt in an end-to-end manner using the gradient descent method.

Chapter 5. Experiments

5.1 Datasets and evaluation methods

We evaluated DSAN performance using several tabular datasets from the UC Irvine Machine Learning Repository. [9] Table 5.1 contains the details. All datasets feature several numerical and categorical variables; all contain over 10,000 records. In the real world, it is difficult to know the correct missing values; therefore, we removed all existing missing values. We then generated missing values that we knew were correct. We evaluated two types of performance, of which the first was imputation performance assessed by calculating the errors between the imputed and original values. For numerical variables, we calculated the normalized root mean squared errors(NRMSEs) [18]; for categorical variables, we derived error rates. The second performance type was downstream task performance. We evaluated the performances of models trained with imputed datasets. The downstream task was a binary classification task; performance was evaluated by deriving area under the curve receiver operating characteristic (AUC-ROC) scores. We used the same classifier (logistic regression) for all cases. We split the data into 80% training and 20% test sets, and created 5 - 20% of "missingness" in the training set using the MCAR(Missing Completely At Random) approach. [19]. All imputation methods (the DSAN and the others) were compared in terms of imputations made for the incomplete training set; we then evaluated imputation performance. The classifier was trained with the imputed training sets;

we then evaluated downstream task performance. To generalize the results, we applied k-fold cross-validation.

Table 5.1: The experimental datasets.

Datasets	The number of Numerical variables	The number of Categorical variables	The number of Records
Adult	9	6	30162
Bank	10	7	11162
Online	10	8	12330
Churn	5	6	10000

5.2 Experimental settings.

We compared DSAN performance to that of a statistical (Mean/Mode) method and that of a classical machine-learning imputation method (MissForest). [21] In the feature representation module, the DSAN featured $d = 16$ column-embedding; the multi-head attention layer had four heads. As mentioned above, the FC unit of the feature representation module had 16 nodes, the same number as that of the embedding dimension d. We dropped out 40% of the input values when denoising. The shared-layer module featured six layers (FC units), and the task-specific module featured layers with nodes appropriate for the various output dimensions. We trained the DSAN using the Adam optimizer [15] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, learning rate $\gamma = 0.003$ and weight decay $\lambda = 10^{-5}$.

5.3 Imputation performance

Table 5.2: Numerical variable imputational performances (NRMSEs).

dataset	Methods	5% missing	10% missing	15% missing	20% missing
Adult	Mean	1.0017	0.9972	1.0002	1.0125
	MissForest	0.8358	0.8561	0.8745	0.9071
	DSAN(Ours)	0.7999	0.8099	0.8247	0.8488
Bank	Mean	0.9562	0.9651	1.0000	0.9778
	MissForest	0.7703	0.8164	0.8586	0.8783
	DSAN(Ours)	0.7580	0.7860	0.8282	0.8226
Online	Mean	1.0012	0.9836	0.9875	0.9947
	MissForest	0.6038	0.6504	0.6915	0.7086
	DSAN(Ours)	0.6311	0.6413	0.6619	0.6738
Churn	Mean	0.9949	1.0052	0.9966	1.0014
	MissForest	0.9850	0.9964	0.9986	1.0208
	DSAN(Ours)	0.9625	0.9725	0.9685	0.9757

Table 5.2 shows the numerical variable imputations; we used $NRMSE$ as the evaluation metric.

$$NRMSE = \sqrt{\frac{\mathbb{E}[(x - \hat{x})^2]}{Var(x)}} \tag{5.1}$$

where x is the true and $\hat{x}$ the imputed value. The $NRMSE$ is the average of the n numerical variables using only the missing values. In general, as the missing rate increases, imputational performance decreases because fewer observed

values are used to predict substituted values. If the model is perfectly imputational, $NRMSE = 0$; if the model imputes the average value for each variable, $NRMSE \approx 1$. Therefore, a lower $NRMSE$ value indicates a better imputation. Table 5.3 shows the average $NRMSE$ values. The DSAN outperformed the other

Table 5.3: Average numerical variable imputational performances.

dataset	Methods	Average	Improvement
Adult	Mean	1.0029	+ 22.18%
	MissForest	0.8684	+ 5.79%
	DSAN(Ours)	**0.8208**	
Bank	Mean	0.9748	+ 22.05%
	MissForest	0.8309	+ 4.03%
	DSAN(Ours)	**0.7987**	
Online	Mean	0.9918	+ 52.10%
	MissForest	0.6636	+ 1.77%
	DSAN(Ours)	**0.6520**	
Churn	Mean	0.9995	+ 3.07%
	MissForest	1.0002	+ 3.13%
	DSAN(Ours)	**0.9698**	

methods in terms of numerical variable imputation. Compared to mean imputation, the performance improvements were 22.18% for the Adult, 22.05% for the Bank, 52.10% for the Online, and 3.07% for the Churn datasets. Compared to

MissForest, the performance improvements were 5.79% for the Adult, 4.03% for the Bank, 1.77% for the Online, and 3.13% for the Churn datasets. For the latter dataset, we confirmed that machine-learning-based regression imputation performance was rather poor when numerical variables exhibited a uniform distribution. Therefore, although performance improved, the improvement was not great. Table 5.4 shows the results of imputation in terms of categorical variables. We used

Table 5.4: Categorical variable imputation performances (error rates).

dataset	Methods	5% missing	10% missing	15% missing	20% missing
Adult	Mode	0.4112	0.4129	0.4134	0.4126
	MissForest	0.2147	0.2238	0.2342	0.2425
	DSAN(Ours)	0.2103	0.2173	0.2239	0.2325
Bank	Mode	0.4063	0.4088	0.4093	0.4074
	MissForest	0.2460	0.2558	0.2674	0.2730
	DSAN(Ours)	0.2501	0.2571	0.2678	0.2732
Online	Mode	0.4166	0.4198	0.4170	0.4186
	MissForest	0.3387	0.3473	0.3539	0.3604
	DSAN(Ours)	0.3366	0.3465	0.3498	0.3569
Churn	Mode	0.4055	0.4060	0.4008	0.4047
	MissForest	0.3561	0.3608	0.3652	0.3770
	DSAN(Ours)	0.3436	0.3576	0.3549	0.3665

$ErrorRate$ as the evaluation metric.

$$ErrorRate = \frac{|x \neq \hat{x}|}{|x|} \tag{5.2}$$

where x is the true and $\hat{x}$ the imputed value. As for $NRMSE$, the $ErrorRate$ is the average of k categorical variables using only the missing values. If the model is perfectly imputational, $ErrorRate = 0$. Thus, a lower $ErrorRate$ indicates better imputational performance. Table 5.5 shows the average $ErrorRate$. Compared

Table 5.5: The average categorical variable imputation performances.

dataset	Methods	Average	Improvement
Adult	Mode	0.4125	+ 86.66%
	MissForest	0.2288	+ 3.53%
	DSAN(Ours)	**0.2210**	
Bank	Mode	0.4080	+ 55.68%
	MissForest	**0.2606**	- 0.57%
	DSAN(Ours)	0.2620	
Online	Mode	0.4180	+ 20.31%
	MissForest	0.3501	+ 0.76%
	DSAN(Ours)	**0.3475**	
Churn	Mode	0.4043	+ 13.67%
	MissForest	0.3648	+ 2.57%
	DSAN(Ours)	**0.3556**	

to mode imputation, the performance improvements were 86.66% for the Adult,

55.68% for the Bank, 20.31% for the Online, and 13.67% for the Churn datasets. Compared to MissForest, the improvements were 3.53% for the Adult, 0.76% for the Online, and 2.57% for the Churn datasets. Thus, the DSAN outperformed the other models in terms of category variable imputation.

5.4 Downstream task performance

To evaluate downstream task performance, we trained a binary classifier (a logistic regressor) with an imputed dataset and evaluated classifier performance using 20% of the test dataset. All experimental datasets featured binary-labeled classes: 'Income' in Adult, 'Deposit' in Bank, 'Revenue' in Online, and 'Exited' in Churn. Table 5.6 shows the downstream task performances using the AUC-ROC score as the evaluation metric; this reveals how well a classifier separates two classes in a dataset. A score closer to 1 indicates a better classifier. The original performance is that after training with the complete dataset but without generating missing values; thus, this is the upper bound of classifier performance. If the performance is similar to that of a classifier trained with the original dataset, the imputed and original datasets are of similar quality. Table 5.7 shows the average downstream task performances. Compared to a classifier trained with the original dataset, the relative performance deteriorations ranged up to 0.1%. This shows that the DSAN-imputed and the original datasets were of similar quality. For the Online dataset, the performance improvement was 0.2% because some noise in the original data was resolved during imputation. In terms of mean/mode imputation, the relative performance improvements were 0.56% for the Adult, 0.37% for the Bank, and 1.55% for the Online datasets. For the

Table 5.6: Results of downstream task performance (AUC-ROC figures).

dataset	Methods	5% missing	10% missing	15% missing	20% missing
Adult	Original		0.9052		
	Mean/Mode	0.9034	0.9015	0.8978	0.8962
	MissForest	0.9049	0.9049	0.9041	0.9036
	DSAN(Ours)	0.9050	0.9050	0.9047	0.9045
Bank	Original		0.9030		
	Mean/Mode	0.9020	0.9002	0.8989	0.8962
	MissForest	0.9030	0.9028	0.9028	0.9019
	DSAN(Ours)	0.9028	0.9029	0.9029	0.9020
Online	Original		0.8945		
	Mean/Mode	0.8889	0.8870	0.8796	0.8765
	MissForest	0.8952	0.8983	0.8980	0.8993
	DSAN(Ours)	0.8951	0.8970	0.8964	0.8984
Churn	Original		0.8325		
	Mean/Mode	0.8324	0.8327	0.8308	0.8317
	MissForest	0.8322	0.8313	0.8298	0.8295
	DSAN(Ours)	0.8323	0.8320	0.8316	0.8307

Table 5.7: Average downstream task performances.

dataset	Methods	Average	Improvement
Adult	Original	0.9052	- 0.00%
	Mean/Mode	0.8997	+ 0.56%
	MissForest	0.9044	+ 0.05%
	DSAN(Ours)	**0.9048**	
Bank	Original	0.9030	- 0.00%
	Mean/Mode	0.8993	+ 0.37%
	MissForest	**0.9026**	+ 0.00%
	DSAN(Ours)	**0.9026**	
Online	Original	0.8945	+ 0.20%
	Mean/Mode	0.8830	+ 1.55%
	MissForest	**0.8977**	- 0.11%
	DSAN(Ours)	0.8967	
Churn	Original	0.8325	- 0.10%
	Mean/Mode	**0.8319**	- 0.03%
	MissForest	0.8307	+ 0.11%
	DSAN(Ours)	0.8316	

MissForest dataset, the performance did not improve.

5.5 Interpreting attention

Unlike MLP-based models, which are difficult to interpret, some results afforded by attention-based models can be interpreted. The DSAN learns the interactions among each variable using a self-attention layer. It is possible to detect the features employed to make decisions. We employed the Adult dataset to explore this topic. This dataset contains individual annual incomes, and various relevant factors. The Adult data include 'relationship' and 'sex' variables. The 'relationship' variable represents a family relationship and can assume values such as 'Husband', 'Wife', and 'Unmarried'. If the 'relationship' variable has a 'Husband' value, the 'sex' variable must be 'Male'. This can serve as the basis for imputation of a value for 'sex' using the 'relationship' variable.

To confirm the learning ability of DSAN, we intend to take the attention weights from the self-attention layer in the feature representation module shown in Fig. 4.1 (A) and to identify what features DSAN focuses on by visualizing the attention weights in the form of a heatmap. The relatively high weighted features can be considered to have contributed more to predicting missing values; that is, analyzing the attention weights of the missing value embedding vector can allow to detect the highly contributed features for the MVI task.

Fig. 5.1 shows the self-attention weights of 20 records when the 'relationship' variable had the 'Husband' value and the 'sex' variable predicted the substituted value 'Male' for the missing value. Each row is the attention weight of the 'sex:NULL' feature of the self-attention layer. It is apparent that the 'sex:NULL'

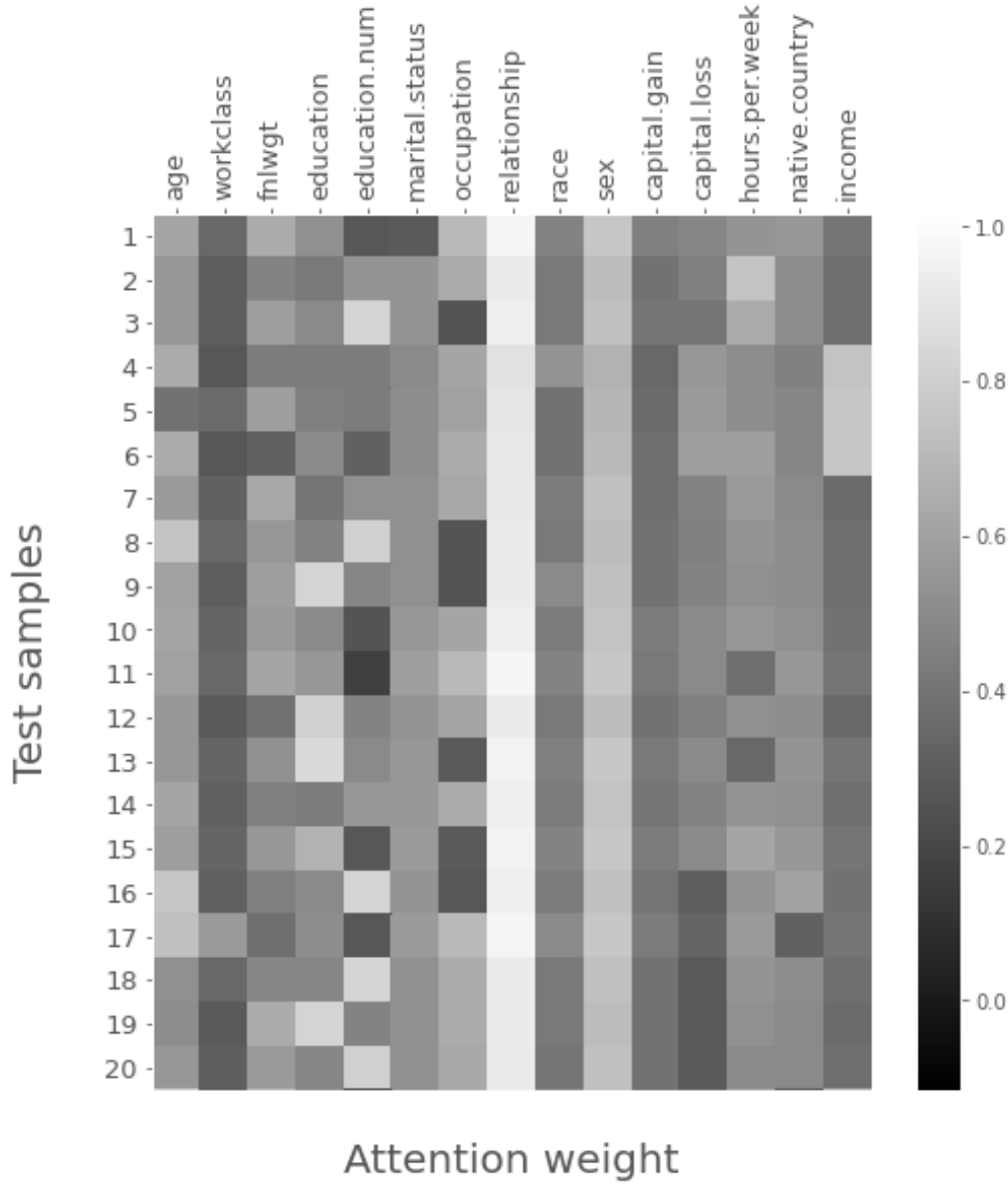

Figure 5.1: Self-attention weights of 20 records of the 'sex:NULL' feature when the 'relationship' variable had the 'Husband' value.

features focus on the ‘relationship:Husband’ feature, indicating that the DSAN can capture the interactions among other variables when predicting substituted values.

Chapter 6. Conclusion

We developed a novel, end-to-end imputational neural network termed the Denoising Self-Attention Network (DSAN) to solve the problem of missing values within tabular data. To verify the model, we downloaded several real-world tabular datasets and evaluated two types of performance, of which one was imputational performance. The $NRMSE$ for numerical variables and the $ErrorRate$ for categorical variables were measured. The DSAN outperformed the other methods in terms of imputation of both numerical and categorical variables. We also explored downstream task performance. We trained a binary classifier using original and imputed datasets, and evaluated performance. A classifier trained with DSAN-imputed data performed similarly to a classifier trained with original data. Thus, the imputed dataset was of similar quality to the original dataset because DSAN constructed a high-quality training dataset. In addition, we offer a visual representation of the attentional weights that reveal the DSAN decision-making process when imputing missing values.

References

[1] S. O. Arık and T. Pfister. Tabnet: Attentive interpretable tabular learning. *arXiv*, 2020.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] G. E. Batista and M. C. Monard. An analysis of four missing data treatment methods for supervised learning. *Applied artificial intelligence*, 17(5-6):519–533, 2003.

[5] F. Biessmann, D. Salinas, S. Schelter, P. Schmidt, and D. Lange. " deep" learning for missing value imputationin tables with non-numerical data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2017–2025, 2018.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[10] L. Gondara and K. Wang. Multiple imputation using deep denoising autoencoders. *arXiv preprint arXiv:1705.02737*, 2017.

[11] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

[14] J. Kim, T. Kim, J.-H. Choi, and J. Choo. End-to-end multi-task learning of missing value imputation and forecasting in time-series data. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 8849–8856. IEEE, 2021.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] X. Lai, X. Wu, and L. Zhang. Autoencoder-based multi-task learning for imputation and classification of incomplete data. *Applied Soft Computing*, 98:106838, 2021.

[17] W.-C. Lin and C.-F. Tsai. Missing value imputation: a review and analysis of the literature (2006–2017). *Artificial Intelligence Review*, 53(2):1487–1509, 2020.

[18] S. Oba, M.-a. Sato, I. Takemasa, M. Monden, K.-i. Matsubara, and S. Ishii. A bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096, 2003.

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

[20] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[21] D. J. Stekhoven and P. Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[23] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[24] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell. Understanding data augmentation for classification: when to warp? In *2016 international*

conference on digital image computing: techniques and applications (DICTA), pages 1–6. IEEE, 2016.

[25] J. Yoon, J. Jordon, and M. Schaar. Gain: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning*, pages 5689–5698. PMLR, 2018.

초 록

최근 데이터 중심 의사결정이 큰 관심을 끌고 있으며, 이에 따라 고품질의 데이터를 필요로 한다. 하지만 실제 데이터는 다양한 이유에 의해 종종 결측값을 포함하며, 데이터 중심 의사결정을 부정확하게 만든다. 이러한 결측값 문제를 해결하기 위해 결측값 보간 기법에 대한 많은 연구가 활발히 수행되었다. 테이블 데이터를 처리하기 위한 기존 머신러닝 기반 결측 데이터 보간 기법은 수치형 변수에만 적용되거나, 변수별로 개별적인 예측 모형을 만들기 때문에 시간이 많이 걸리고 번거롭다. 이에 본 논문은 수치형, 범주형 변수가 혼합된 테이블 데이터에 적용 가능한 데이터 보간 신경망인 Denoising Self-Attention Network(DSAN) 를 제안한다. DSAN은 셀프 어텐션과 디노이징 기법을 결합하여 견고한 특징 표현 벡터를 학습하고, 멀티태스크 러닝을 통해 여러 적절한 대체값을 병렬적으로 예측할 수 있다. 제안 모델의 유효성을 검증하기 위해 여러 실제 테이블 데이터에 대하여 결측값을 임의로 생성한 후 데이터 보간 실험을 수행했다. 그런 다음 보간 성능 및 후속 작업 성능을 평가하여 모델의 우수성을 보인다.

핵 심 낱 말 머신러닝, 딥러닝, 데이터 품질, 결측값, 데이터 정제, 어텐션