Spatio-Temporal Missing Data Imputation for Smart Power Grids

Sanmukh R. Kuppannagari
kuppanna@usc.edu
Department of Electrical and Computer Engineering,
University of Southern California
Los Angeles, California

Yao Fu
yaof@usc.edu
Department of Electrical and Computer Engineering,
University of Southern California
Los Angeles, California

Chung Ming Chueng
chungmin@usc.edu
Department of Computer Science,
University of Southern California
Los Angeles, California

Viktor K. Prasanna
prasanna@usc.edu
Department of Electrical and Computer Engineering,
University of Southern California
Los Angeles, California

ABSTRACT

Availability of high fidelity timeseries data is imperative for critical power grid operational tasks such as state estimation, DER scheduling, etc. However, the data obtained from the metering infrastructure is prone to disruptions due to communication outages leading to missing values. State-of-the-art smart power grid Missing Data Imputation (MDI) algorithms either operate on individual timeseries and are unable to capture spatial dependencies due to the power grid topology or they operate on the entire dataset, requiring complex models which lead to overfitting.

In this work, we develop a novel technique to perform spatio-temporal missing data imputation. Using the power grid topology and timeseries data obtained from the metering infrastructure in the grid as input, we develop a Spatial-Temporal Graph Neural Network based Denoising Autoencoder (STGNN-DAE) that performs MDI by accounting for both temporal and spatial correlations. Using a real dataset obtained from a distribution test grid in Midwest, Iowa, we compare the proposed model with existing solutions for MDI. We show that our GNN based autoencoder obtains an improvement of 4.3% to 25.2% in error metrics such as Mean Absolute Error compared to the state-of-the-art missing data imputation methods.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Hardware** → *Smart grid*.

KEYWORDS

smart grids, missing data imputation, spatio-temporal deep learning, autoencoders

e-Energy '21, June 28–July 2, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8333-2/21/06.
https://doi.org/10.1145/3447555.3466586

ACM Reference Format:
Sanmukh R. Kuppannagari, Yao Fu, Chung Ming Chueng, and Viktor K. Prasanna. 2021. Spatio-Temporal Missing Data Imputation for Smart Power Grids. In *The Twelfth ACM International Conference on Future Energy Systems (e-Energy '21), June 28–July 2, 2021, Virtual Event, Italy*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3447555.3466586

1 INTRODUCTION

Operations such as estimation of the state variables (voltage, phase angle, etc.), load/generation prediction, scheduling of generators or Distributed Energy Resources (DER), load or solar curtailment etc. are critical to smooth operation of power grids [3, 11, 12, 20, 24, 26]. The efficacy of these operations, and hence the grid stability [26], is heavily dependent upon the quality of the real-time data obtained from the metering infrastructure in the grid.

Modern metering infrastructure such as AMI, SCADA, etc. enable collection of fine-grained high frequency data in real-time [17]. However, they are prone to disruptions due to communication outages, unexpected meter shutdowns, measuring errors, etc. [8] Such disruptions lead to missing values in the dataset. The missing values have the potential to significantly impact the effectiveness (accuracy, optimality, etc.) of the downstream operations and may even lead to catastrophic failures such as supply demand mismatch due to improper scheduling or voltage violations due to improper regulation.

In power grids, the underlying topology leads to strong spatial correlations between the behaviors of the assets of the grid. For example, a $\approx$ 3% voltage increase at a transformer will increase the power consumption of all the loads connected to it by $\approx$ 6%, assuming resistive only load. Thus, modeling these spatial correlations in addition to temporal dependencies can improve the overall accuracy of any data-driven model for the power grid.

State-of-the-art smart power grid Missing Data Imputation (MDI) algorithms operate on individual timeseries [18, 19] or take the entire (or a large portion of) the dataset and as input to perform imputation [4, 22]. The former solutions do not capture spatial dependencies while the latter ones require complex models leading to overfitting and therefore, reduced accuracy. Recently, works such

as [1, 10] have developed analytical models to estimate missing data using information from the neighbors — thus capturing spatial correlations. However, they rely on information such as impedances, current flow, etc. whose reliable availability is difficult.

Recently, Graph Neural Networks (GNN) have gained popularity as a technique to develop data driven deep learning models for domains with an underlying graph structure [27]. The key idea behind these models is that the connection between two consecutive layers of the neural network are modeled based on the underlying graph structure, instead of the all-to-all connections of traditional neural networks. Thus, the information flow between layers follows the physical connections of the underlying power grid topology and the number of layers models the number of hops of neighbors to consider. In the context of power grid, the nodes of the GNN can be used to represent the grid assets such as loads, buses, DERs, etc. while the connections can be used to represent the feeder lines (wires that carry electricity).

In this work, we develop a data-driven model for Spatio-Temporal missing data imputation. Using the power grid topology and time-series data obtained from each meter in the grid as input, we develop a Spatial-Temporal Graph Neural Network based Denoising Autoencoder (STGNN-DAE) that performs MDI by accounting for both temporal and spatial correlations. Using a real dataset obtained from a distribution test grid in Midwest, Iowa, we compare the proposed model with existing solutions for MDI. We obtain an improvement of 4.3% to 25.2% in error metrics such as Mean Absolute Error compared to the state-of-the-art methods for missing data imputation.

2 RELATED WORKS

Several works have explored ways to perform missing data imputation. Works such as [4, 16] include matrix operation and machine learning based models which operate on the entire input matrix to perform imputation. For example, the denoising autoencoder developed in [4] takes the entire matrix with corrupt entries as input to reconstruct an uncorrupted version. The scalability (at deployment time) of the matrix operation based models is limited while the machine learning based models suffer from sample inefficiency as they try to model all-to-all correlations. On the other hand, works such as [9, 14, 18, 19] work on individual time series data and may not be able to capture the spatial correlations. For example, the autoencoders developed in [14, 19] take the daily load profile as input to perform imputation. These works differ from each other based on the choice of the neural network structure used in the autoencoders, [19] used variational autoencoders, while [14] used LSTM networks. [9] uses a k-nearest neighbor based approach to search for past situations similar to the current time series with missing values and uses that to estimate the missing values. Authors in [21] develop a GNN based technique for Missing Data Imputation. However, their technique is not tailored towards time series data and thus does not capture temporal dependencies.

Recently, works such as [1, 10] have developed analytical models to estimate missing data using information from the neighbors — thus capturing spatial correlations. Our proposed technique is a data-driven version of these methods which only needs the connectivity information and forgoes other hard to obtain information such as impedances, current flows, etc.

3 PROBLEM DEFINITION

3.1 Problem of Missing Data in Smart Grids

Smart grids consist of assets such as load consumers, electricity producers, and grid management infrastructure. At any given moment, the matching of the supply of electricity with the demand is critical to ensure smooth operations of the grid [17]. To enable this, grid operators need to constantly monitor the activity of the asset (power injected or consumed) in the grid and take mitigating actions (votage regulation, supply scheduling, etc.) when necessary. This requires accurate and timely data collection from the grid. The task of data collection is performed by the Advanced Metering Infrastructure (AMI) [6]. AMI meters are connected to every asset that needs to be monitored. These meters record the real and reactive power consumed over time by these assets. The meters then transmit the meter readings for every fixed interval through the communication network to a database storage [6].

However, the presence of missing entries in the collected data is a key challenge. Missing data occurs due the following reasons: (a) Intermittent failure of one or a sub-set of AMI meters or the communication network - This introduces single or short sequence of missing entries. (b) Prolonged failure of communication network of AMI meters - This leads to blocks of missing values. A study carried out by [19] shows that in one particular dataset containing about 5 million 15-minute intervals of data, approximately $420k$ intervals are missing. Within these missing intervals, 47.14% are single missing entries while 51.26% are missing blocks under half a day length and about 1% are longer than half a day. These statistics are shown in Table 1.

Table 1: Statistics of missing values borrowed from [19]

Type	Period	Intervals No.	Percentage
Single Entry	1 missing interval	1	47.14%
Block	< half day	(1,12]	51.26%
Block	< one day	(12,24]	1.05%
Block	< one week	(24,192]	0.53%
Block	> one week	(192	0.02%

Presence of missing values has a severe impact on downstream applications which are critical to grid operations such as load forecasting or optimal power flow which rely on the fidelity of the data [15]. For example, for an artificial neural network trained on a complete data, testing on a dataset with less than even 10% of missing data led to a significant drop in the ROC curve [15]. Therefore, development of accurate missing data imputation methods will significantly improve the reliability of grid operations.

3.2 Problem Definition

The missing data imputation problem is to find a model that is able to estimate missing entries in a given dataset. Suppose $\mathbf{y} = \{y_1, y_2, ..., y_T\}$ is a timeseries of length T with missing entries in a set of time intervals $\mathbf{M}$, that is, y_t is missing for $t \in \mathbf{M}$. A missing

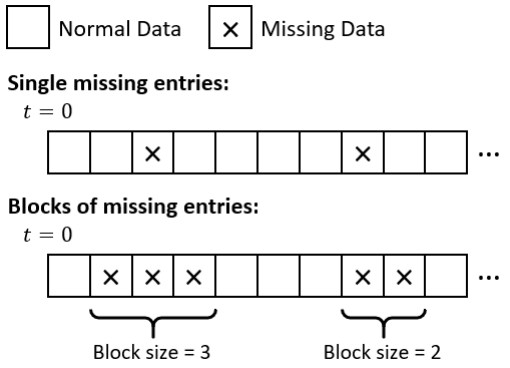

Figure 1: Illustration of time series with single missing entries and block missing entries

data imputation model is a function g on $\mathbf{y}$ such that $\tilde{\mathbf{y}} = g(\mathbf{y})$, where $\tilde{\mathbf{y}}$ has its missing entries in time intervals $\mathbf{M}$ recovered.

We can train a missing data imputation model as follows. Given a timeseries of length T denoted as $\mathbf{x} = \{x_1, x_2, ..., x_T\}$ with no missing data. We want to use $\mathbf{x}$ as training data to train a function f that replaces missing values in another timeseries $\hat{\mathbf{x}}$ of length $\hat{T}$. We denote $\tilde{\mathbf{x}} = f(\mathbf{x}, \hat{\mathbf{x}})$ as $\hat{\mathbf{x}}$ with its missing values replaced using the trained function f. The objective is then to minimize an error function $E(\mathbf{x}, \tilde{\mathbf{x}})$ that defines the difference between the two timeseries.

Missing data in $\hat{\mathbf{x}}$ can either be single missing entries or blocks of missing entries. Fig. 1 illustrates the two types of missing data, where the × symbol represents missing data in the timeseries. Single missing entries refer to a single interval of missing points between intervals with known values. Blocks of missing entries refers to consecutive chunks of missing data. The length of this consecutive block is called the block size. We define "missingness" as a measure of the percentage of data in $\hat{\mathbf{x}}$ that is missing. For example, a missingness of 1% means 1% of the data is missing.

In the context of smart grids, $\mathbf{x}$ is typically a large historically collected timeseries dataset which is used to train the model f. $\hat{\mathbf{x}}$ is typically a recently collected timeseries dataset which needs to be pre-processed before using it as an input in a downstream grid management task such as load forecasting or optimal power flow calculation.

We also make the following assumptions when solving the problem in this paper:

(1) The datasets $\mathbf{x}$ and $\hat{\mathbf{x}}$ are timeseries in nature.
(2) $\mathbf{x}$ does not have any missing entries.

4 METHODOLOGY

In this work, we develop a Spatial-Temporal Graph Neural Network based Denoising Autoencoder (STGNN-DAE) that performs MDI by accounting for both temporal correlations due to the timeseries of the dataset and spatial correlations imposed due to the power grid topology. We first describe the STGNN-DAE model and then describe how it is applied to the problem of Missing Data Imputation (MDI).

4.1 Spatial-Temporal GNN based Auto-Encoder (STGNN-DAE)

An autoencoder is a neural network architecture that aims to learn a concise representation of input data. The output of the autoencoder is a representation of the input data with significantly less dimensions. An autoencoder consists of two components, the encoder $e(\mathbf{x})$ and decoder $d(\mathbf{x})$. The encoder should reduce the dimension of the input data, while the decoder attempts to recover the original data from the reduced data. Denoting the recovered data $\hat{\mathbf{x}} = d(e(\mathbf{x}))$, the network is trained by minimizing the error between $\mathbf{x}$ and $\hat{\mathbf{x}}$. However, there is a problem with autoencoders where the identity function is learned, that is, the network simply maps data points to the reduced form and vice versa. This is prevented by introducing noise to the input $\mathbf{x}$ during training, this variation is called denoising autoencoder.

A Graph Convolution Layer performs convolution on a graph. This convolution aggregate data points of neighboring nodes. As a result, there is a passage of information between neighbors of the graph. Mathematically, it can be written as:

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \tag{1}$$

where $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times D_l}$ denotes the l^{th} layer of the network and $\mathbf{W}^{(l)}$ is the weight parameters of the l^{th} layer. $\mathbf{A}$ denotes the adjacency matrix of the graph g, which is used to compute $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ by adding self-connections to $\mathbf{A}$. $\mathbf{I}_N$ is the identity matrix. $\tilde{\mathbf{D}}$ is the degree matrix calculated by $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$. This kind of layer is useful for data where data points on the graph are more likely to be correlated if they are close neighbors, as it can learn these spatial correlations.

To capture temporal correlations, that is, correlations among values of intervals of a customer that happen near in time, gated 1D convolutional layers can be used. A 1D filter of length K is used to compute a convolution operation on each daily load profile timeseries $\mathbf{x}$ of length T of each customer. Denote the filter as $\Theta \in \mathbb{R}^K$ and the convolution operation as $*$. $\Theta * \mathbf{x}$ performs convolution on $\mathbf{x}$ such that its length is reduced from $\mathbb{R}^l$ to $\mathbb{R}^{(l-K+1)}$. A Gated Linear Unit (GLU) is then used as the activation, which is defined as

$$GLU = (\Theta * \mathbf{x}) \circ \sigma(\Theta * \mathbf{x}) \tag{2}$$

where $\circ$ is the element-wise product operator and σ is the sigmoid function.

In our model, we make use of Spatial-Temporal Blocks (ST Blocks) [25], the structure consists of a gated convolutional layer followed by a graph convolutional layer then one more gated convolutional layer. Both spatial correlations and temporal correlations are learned from these ST Blocks. This is illustrated in Fig. 2. To capture both spatial and temporal correlations of the data with an autoencoder, we propose a Spatial-Temporal Graph Neural Network (ST-GNN) based autoencoder architecture. Both the encoder function $e(x)$ and decoder function $d(x)$ are each an ST Block.

4.2 STGCN-DAE based MDI

In this section, we describe the training workflow for our proposed STGNN-DAE model for Missing Data Imputation (MDI). First, we explain how a DAE is applied for MDI. Given a daily load timeseries

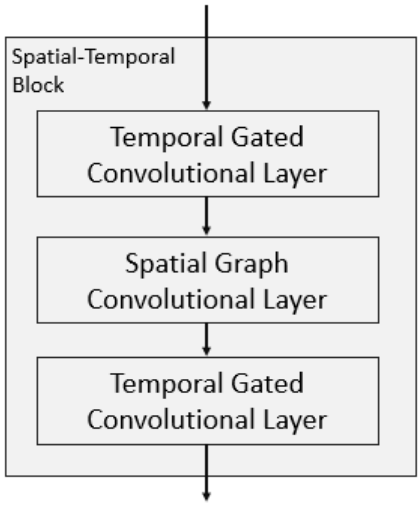

Figure 2: Structure of a Spatial-Temporal Block

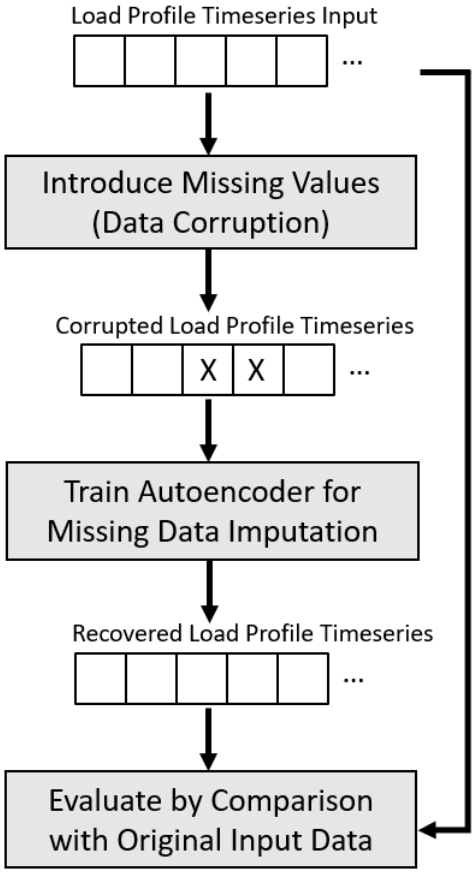

Figure 3: Overall training workflow

Algorithm 1: Autoencoder Training Workflow

Input: $\mathbf{x}$ - Input timeseries
Calculate corrupted data $\hat{\mathbf{x}}$ from $\mathbf{x}$
For each epoch:
  Compute output through autoencoder $\tilde{\mathbf{x}} = d(e(\hat{\mathbf{x}}))$
  Compute error function $E(\mathbf{x}, \tilde{\mathbf{x}})$
  Backpropagate error to update autoencoder parameters

with no missing data $\mathbf{x}$, it is first corrupted by replacing some of its missing values to form $\hat{\mathbf{x}}$. Next, we define an encoder $e(\mathbf{x})$ and decoder $d(\mathbf{x})$ such that it outputs $\tilde{\mathbf{x}} = d(e(\hat{\mathbf{x}}))$. The encoder and decoder are trained by minimizing the error between the ground truth $\mathbf{x}$ and the output $\tilde{\mathbf{x}}$. The overall workflow is summarized in Figure 3 and Algorithm 1.

As described in Section 4.1, the encoder and decoder functions in STGNN-DAE is an ST Block structure. Since the graph convolution layer performs convolution on all nodes connected via graph edges, the input needs to contain timeseries from all nodes. Therefore, the input $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ is the collection of all daily loads from all customers where $\mathbf{x}_i$ is the daily load of customer i and N is the total number of customers.

4.2.1 Data preprocessing. To prepare the dataset for MDI model training, we need to introduce missing values in each daily load timeseries of each customer for both real and reactive powers. Missing values are either introduced by single intervals or by blocks following the observations in Table 1. We define "missingness" as the percentage of missing values in each timeseries. For single missing intervals, a set of intervals is picked randomly so that the number of intervals picked is equal to the missingness percentage of the total number of intervals. These picked intervals are then replaced as missing values. For blocks, we first pick a block size B and then a set of intervals according to the missingness just like single intervals. Afterwards, each picked interval and $B-1$ intervals following it are replaced as missing values. This creates blocks of missing data similar to the bottom example in Fig. 1.

4.2.2 Missing Data Configurations. We evaluate MDI models based on the comparison of their performance on 9 different missing data configurations. We use 3 single missing intervals configurations and 6 block missing configurations. Each configuration differs by the missingness and block size choice. The configurations is summarized in Table 6.

Table 2: Missing Data Configurations

Config.	Type	Missingness	Block Size
C1	Single	0.2	-
C2	Single	0.3	-
C3	Single	0.5	-
C4	Block	0.45	3
C5	Block	0.6	3
C6	Block	0.4	4
C7	Block	0.6	4
C8	Block	0.48	8
C9	Block	0.64	8

4.3 Deployment of STGCN-DAE for MDI

In Section 4.2.2, we train a model using only a single type of missing data configuration (Table 6). Thus, models that are trained and evaluated in each configuration may be specialized in MDI for each specific missing data type. However, in real-world the dataset may consist of several different missing data configurations and using a model trained on a single configuration may lead to reduced accuracy. To mitigate this issue, a potential deployment scenario for missing data imputation is as follows:

- Train an ST-GCN model for each configuration described in Table 6.
- When a dataset $\hat{\mathbf{x}}$ needs to be process for missing data imputation, do the following:
  (1) Identify the blocks of missing data.
  (2) For each block, select the model whose block sizes is the closest to the size of the missing block.

However, as we show in Section 5.5.2, a single ST-GCN model is able to obtain high accuracy on a wide range of missingness configurations. Thus, in practice only one (or a few) model which obtains high accuracy over most of the configurations described in Table 6 can be chosen for deployment.

5 EXPERIMENTS AND RESULTS

5.1 Baselines

We consider the following Missing Data Imputation models as baseline for comparison.

5.1.1 Linear Interpolation (LI). Linear Interpolation is a data imputation method that fits a linear model using the two values from before and after each missing data block [18]. Each value within the missing data block is then estimated from the linear model.

Mathematically, we can express the method as follows. Given a timeseries $\mathbf{x} = \{x_1, x_2, ..., x_T\}$ in which the values between intervals t to $t + W - 1$ are missing, where W is the missing block size, the missing values are computed by the equation below.

$$x_{t+w} = x_{t-1} + (x_{t+W} - x_{t-1}) * \frac{W - w}{W} \tag{3}$$

The advantage of this method is that it is quick and deterministic. The disadvantage is that the model is very simple and cannot model complicated timeseries trends, especially when the missing period is long.

5.1.2 Historical Average (HA). Historical Average makes use of the intuition that timeseries data in smart grids has high regularity, that is, it tends to show similarity over same times of the days, or same days of the week and so on. We use the implementation from [18] that imputes missing values by taking the average of intervals that are within a certain range with respect to the missing value timestamp (defined as [Day, Hour of day, Minute of hour]). For the baseline in this paper, we use the range [±8 day, ± 1 hour, and ±1 minute]. This means that any interval which the timestamp is within the defined range of the considered timestamp of the missing interval, it will be included in the set of intervals used for computing the HA.

In addition to these traditional methods, we use Denoising Autoencoder [23] based models which have been recently proposed for Missing Data Imputation. These models learn to reconstruct original dataset from the corrupted dataset. We consider the following three versions of denoising auto-encoders.

5.1.3 ConvNN-DAE. A ConvNN-DAE is a Denoising Autoencoder that only uses convolutional layers and fully connected layers in the network architecture [19]. Authors in [19] develop a ConvNN-DAE which consists of three layers of 1D convolution on a single timeseries input, followed by a fully connected layer, followed by three layers of 1D transposed convolution. For a dataset with multiple nodes, the authors train a single model which takes as input a single daily load timeseries data, i.e. timeseries data corresponding to 24 hour interval of a day for a single node, and performs missing data imputation on the same. In other words, a single model is used to perform missing data imputation for all the nodes of the grid.

5.1.4 Multiple Imputations Denoising Autoencoder (MIDA). Multiple Imputations Denoising Autoencoder is also a denoising autoencoder approach [5] that uses only fully connected layers. Unlike the other DAEs used in the baselines, the encoder function of MIDA maps the inputs to a higher dimension, and the decoder function maps the higher dimension representation back to the original input. The model is trained while corrupting the inputs by randomly removing certain values randomly for each iteration.

5.1.5 LSTM. We implemented the Long Short Term Memory(LSTM)-based DAE propoced in [13]. In this DAE architecture, both encoder and decoder functions are LSTM recurrent neural networks [7]. Recurrent neural networks (RNN) are neural networks that use intermediate outputs at each layer as the input for the next step in the sequence, this improves the abilitiy of the network to learn correlations over sequential data. LSTM cells are popular constructions used in RNNs as it solves the problem of vashing gradient by using a forget gate structure. By using LSTM networks in the autoencoder, it allows the autoencoder to specialize in analyzing sequential data like timeseries.

5.2 Datasets

We use a dataset with data from 240 customers from Iowa covering 365 days [2]. The dataset contains real power consumption measurements of each customer in 1-hour intervals. Additionally, it contains simulated reactive power consumption of the customers. The topology of the network is also provided in the dataset. We split the dataset into training, validation, and testing as follows: day $1 - 275$ for training, day $276 - 305$ for validation and day $306 - 365$ for testing.

5.3 Experimental Setup

For each baseline, we perform hyperparameter searching to find the best model configuration. We train models by varying their hyperparameters with the training dataset. We then evaluate each model using the validation dataset. We choose the model that obtains lowest validation error and test the model using the testing dataset to compute the errors. The errors on the testing dataset are reported.

We use the six missing data imputation models described in Section 5.1 as baselines in addition to our proposed STGNN-DAE model. Each baseline is tested on nine different test datasets. The nine different datasets are obtained by corrupting the original dataset corrupted with the nine different configurations as shown in Table 6.

5.4 Evaluation Metrics

We evaluate the MDI results using three different error metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Normalized Absolute Error (NAE) [19]. When calculating the error, we only consider the missing intervals of customer load timeseries in the testing dataset, that is, the error between recovered missing values with MDI compared to the ground truth. Denote C as the set of customers in the testing dataset, T as the total number of time intervals, $\tilde{x}_{c,t}$ as the output of MDI for customer c at time t, and $y_{c,t}$ as the ground truth value for customer c at time t. A binary matrix $\mathbf{M}$ indicates if interval t for customer c is a missing value interval by setting $M_{c,t} = 1$, and otherwise $M_{c,t} = 0$. Each error metric is defined as follows:

$$MAE = \frac{1}{N}\sum_{T}^{t=0}\sum_{c\in C} M_{c,t}|\tilde{x}_{c,t} - y_{c,t}| \tag{4}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{T}^{t=0}\sum_{c\in C} M_{c,t}||\tilde{x}_{c,t} - y_{c,t}||_2^2} \tag{5}$$

$$NAE = \frac{1}{N}\sum_{T}^{t=0}\sum_{c\in C} M_{c,t}\frac{|\tilde{x}_{c,t} - y_{c,t}|}{X_i} \tag{6}$$

Where N is the total number of missing intervals and X_i is the average load.

The errors in terms of each of the 3 metrics are computed for each of the 9 configurations in Tab. 6.

5.5 Results and Discussion

5.5.1 Comparison with State-of-the-art MDI Models. Tables 3-5 summarize the experiment results of all methods with respect to the three evaluation metrics MAE, RMSE, and NAE, respectively.

As evident from the table, STGNN-DAE outperforms all the algorithms in terms of MAE, RMSE and NAE for most of the configurations. The only exception is the configuration 'C1' where Linear Interpolation (LI) performs the best. In other words, for single and sparse missing entries, simple linear interpolation using neighboring values gives the best performance.

Specifically, the performance improvement in terms of the error metrics range from 4.3% to 25.2% against the second best performing model. LSTM and MIDA models performed the second best on most of the configurations. MIDA tries to capture all to all spatial correlations leading to overfitting of the model. On the other hand, LSTM trains using only individual timeseries data, and fails to consider spatial correlations. Thus, by using a model which captures only the relevant correlations, STGNN-DAE able to achieve the best accuracy.

5.5.2 Performance of STGNN-DAE over Varying Block Sizes. To further test the capability of STGNN-DAE in a scenario closer to real life data, as described in Section 4.3, we test a STGNN-DAE trained on configuration 'C3' on a wide range of missingness configurations and block sizes. First, we test the trained STGNN-DAE on configurations that include a wider range of block sizes that spans between 2 and 12. The results are shown in Tables 7 and 7. In terms of MAE, RMSE and NAE, the range of error over the full range of configurations are $[1.2227, 1.8367]$, $[3.1722, 5.2271]$ and $[0.3378, 0.4611]$ respectively. We can see that despite the wider range of block sizes, the errors of the model reside within a small range that is similar to when a model is trained for each configuration. Next, we test the trained STGNN-DAE on datasets with mixed block sizes. We define 7 different datasets where the block size for each missing period is sampled from a Gaussian distribution with varying mean and variance of 9. Case numbers $2-8$ of Table 9 describes these 7 datasets. Case 1 is a dataset that randomly samples distributions of case $2-8$ with equal probability. Similar to the varying configurations experiment, we see that the errors of the trained model reside within a small range despite the varying block sizes. In terms of MAE, RMSE and NAE, the range of error over the full range of test case datasets are $[1.5192, 1.8727]$, $[3.7601, 5.1375]$ and $[0.3658, 0.4506]$ respectively. In particular, even when the block sizes vary greatly in case 1, the model was able to perform satisfactorily.

Overall, we can see that STGNN-DAE adapts well to most missing block configurations. This is crucial as in the real world, missing data have varying block sizes of sizes as shown in Table 1.

6 ACKNOWLEDGEMENTS

This work has been sponsored by the U.S. Army Research Office (ARO) under award number W911NF1910362 and the U.S. National Science Foundation (NSF) under award numbers 1911229 and 2009057.

7 CONCLUSION

In this work, we developed a data-driven model for spatio-temporal missing data imputation. Using the power grid topology and time-series data obtained from each meter in the grid as input, our Spatial-Temporal Graph Neural Network based Denoising Autoencoder (STGNN-DAE) performs MDI by accounting for both temporal and spatial correlations. We compared our proposed model with existing solutions for MDI using a real dataset obtained from a distribution test grid in Midwest, Iowa and obtained 4.53-25.2% improvement compared to the state-of-the-art methods on varying missing data configurations.

REFERENCES

[1] Cruz E Borges, Oihane Kamara-Esteban, Tony Castillo-Calzadilla, Cristina Martin, and Ainhoa Alonso-Vicario. 2020. Enhancing the missing data imputation of primary substation load demand records. *Sustainable Energy, Grids and Networks* (2020), 100369.

[2] Fankun Bu, Yuxuan Yuan, Zhaoyu Wang, Kaveh Dehghanpour, and Anne Kimber. 2019. A time-series distribution test system based on real utility data. In *2019 North American Power Symposium (NAPS)*. IEEE, 1–6.

[3] Yohwan Choi, Seunghyoung Ryu, Kyungnam Park, and Hongseok Kim. 2019. Machine learning-based lithium-ion battery capacity estimation exploiting multi-channel charging profiles. *IEEE Access* 7 (2019), 75143–75152.

[4] L Gondara and K Wang. [n.d.]. Multiple imputation using deep denoising autoencoders. arXiv 2017. *arXiv preprint arXiv:1705.02737* ([n. d.]).

Table 3: Mean Absolute Error of Missing Data Imputation

Configuration	C1	C2	C3	C4	C5	C6	C7	C8	C9
Missingness	0.2	0.3	0.5	0.45	0.6	0.4	0.6	0.40	0.64
Block Size	1	1	1	3	3	4	4	8	8
LI	**1.2956**	1.9576	2.3023	3.2338	3.3315	3.3837	3.5532	3.9385	3.9691
HA	1.4561	2.0594	2.3813	2.6409	2.8305	2.5601	2.9503	2.8734	3.1301
ConvNN-DAE	1.9952	2.1569	2.3344	2.1307	2.2741	2.0391	2.2847	2.2308	2.4001
MIDA	1.6619	1.7862	1.8922	1.8371	1.9465	1.7747	1.9831	1.9505	2.0705
LSTM	1.4345	1.5669	1.6231	1.7022	2.3525	1.7043	1.8196	1.8948	1.9861
STGNN-DAE	1.3662	**1.4594**	**1.3535**	**1.4631**	**1.5495**	**1.5445**	**1.7689**	**1.7448**	**1.8418**

Table 4: Root Mean Squared Error of Missing Data Imputation

Configuration	C1	C2	C3	C4	C5	C6	C7	C8	C9
Missingness	0.2	0.3	0.5	0.45	0.6	0.4	0.6	0.40	0.64
Block Size	1	1	1	3	3	4	4	8	8
LI	3.7498	5.7139	6.5108	8.4125	8.5543	8.6865	9.1174	9.9091	9.9311
HA	4.1879	5.9543	6.6996	7.2747	7.5796	7.0681	7.9281	7.8868	8.4297
ConvNN-DAE	7.219	7.6038	7.8761	7.6015	7.7367	7.3871	7.8773	8.0657	8.2215
MIDA	4.6821	5.1442	5.5111	5.1757	5.3988	4.9191	5.5619	5.4771	5.8306
LSTM	3.5833	4.1787	4.0594	4.3078	6.5236	4.3057	4.9967	**5.0051**	5.4152
STGNN-DAE	**3.5614**	**3.7361**	**3.5449**	**3.8142**	**4.0371**	**4.1188**	**4.7684**	5.0303	**5.0196**

Table 5: Normalized Absolute Error of Missing Data Imputation

Configuration	C1	C2	C3	C4	C5	C6	C7	C8	C9
Missingness	0.2	0.3	0.5	0.45	0.6	0.4	0.6	0.40	0.64
Block Size	1	1	1	3	3	4	4	8	8
LI	**0.3119**	0.4708	0.5538	0.7774	0.8009	0.8139	0.8541	0.9465	0.9542
HA	0.3517	0.4961	0.5734	0.6353	0.6809	0.6164	0.7094	0.6904	0.7526
ConvNN-DAE	0.4793	0.5177	0.5604	0.5113	0.5456	0.4901	0.5482	0.5355	0.5757
MIDA	0.3994	0.4291	0.4546	0.4413	0.4684	0.4263	0.4762	0.4681	0.4975
LSTM	0.3451	0.3771	0.3908	0.4098	0.5652	0.4102	0.4379	0.4558	0.4781
STGNN-DAE	0.3287	**0.3514**	**0.3261**	**0.3523**	**0.3728**	**0.3719**	**0.4254**	**0.4197**	**0.4435**

Table 6: Real Mean Value during each experiment execution

Config	Missingness	Block Size	Real Mean Value
C1	0.2	-	4.1852
C2	0.3	-	4.1787
C3	0.5	-	4.1762
C4	0.45	3	4.2184
C5	0.6	3	4.2068
C6	0.4	4	4.2269
C7	0.6	4	4.2376
C8	0.40	8	4.4457
C9	0.48	8	4.4113

[5] Lovedeep Gondara and Ke Wang. 2018. Mida: Multiple imputation using denoising autoencoders. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 260–272.

[6] David G Hart. 2008. Using AMI to realize the Smart Grid. In *2008 IEEE Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century*, Vol. 10. sn.

[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[8] Zhen Jiang, Di Shi, Xiaobin Guo, Guangyue Xu, Li Yu, and Chaoyang Jing. 2018. Robust smart meter data analytics using smoothed ALS and dynamic time warping. *Energies* 11, 6 (2018), 1401.

[9] Minkyung Kim, Sangdon Park, Joohyung Lee, Yongjae Joo, and Jun Kyun Choi. 2017. Learning-based adaptive imputation methodwith kNN algorithm for missing power data. *Energies* 10, 10 (2017), 1668.

[10] Daisuke Kodaira and Sekyung Han. 2018. Topology-based estimation of missing smart meter readings. *Energies* 11, 1 (2018), 224.

[11] Sanmukh Kuppannagari, Rajgopal Kannan, and Viktor K Prasanna. 2019. Approximate Scheduling of DERs with Discrete Complex Injections. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*. 204–214.

[12] Sanmukh R Kuppannagari, Rajgopal Kannan, and Viktor K Prasanna. 2017. Optimal net-load balancing in smart grids with high PV penetration. In *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*. 1–10.

[13] You Lin, Jianhui Wang, and Mingjian Cui. 2019. Reconstruction of power system measurements based on enhanced denoising autoencoder. In *2019 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 1–5.

[14] Jun Ma, Jack CP Cheng, Feifeng Jiang, Weiwei Chen, Mingzhu Wang, and Chong Zhai. 2020. A bi-directional missing data imputation scheme based on LSTM and transfer learning for building energy data. *Energy and Buildings* (2020), 109941.

[15] Mia K Markey, Georgia D Tourassi, Michael Margolis, and David M DeLong. 2006. Impact of missing data in evaluating artificial neural networks trained on

Table 7: Testing STGNN-DAE on Varying Missingness Configurations - Block Sizes 2 - 6

Missingness	0.2	0.5	0.2	0.5	0.18	0.48
Block Size	2	2	5	5	6	6
MAE	1.2227	1.3793	1.3053	1.5208	1.3498	1.5334
RMSE	3.1722	3.5953	3.4357	4.2383	3.5492	4.2457
NAE	0.3378	0.3945	0.3787	0.3964	0.3853	0.3992
Real Mean Value	4.1911	4.1848	4.2385	4.2594	4.3067	4.2985

Table 8: Testing STGNN-DAE on Varying Missingness Configurations - Block Sizes 7 - 12

Missingness	0.21	0.49	0.27	0.54	0.25	0.5	0.25	0.48
Block Size	7	7	9	9	10	10	12	12
MAE	1.4008	1.6675	1.5043	1.7173	1.5615	1.8367	1.6845	1.6845
RMSE	3.7114	4.5431	4.1391	4.8386	4.4224	5.2271	4.9303	4.9303
NAE	0.3951	0.4251	0.4107	0.4382	0.4204	0.4611	0.4418	0.4418
Real Mean Value	4.3792	4.3557	4.5096	4.5013	4.5701	4.5556	4.6808	4.6808

Table 9: Testing STGNN-DAE on Mixed Block Sizes

Case No.	1	2	3	4	5	6	7	8
Block Size	Random	$\mathcal{N}(2,9)$	$\mathcal{N}(5,9)$	$\mathcal{N}(6,9)$	$\mathcal{N}(7,9)$	$\mathcal{N}(9,9)$	$\mathcal{N}(10,9)$	$\mathcal{N}(12,9)$
MAE	1.6251	1.5192	1.6111	1.6279	1.7356	1.7528	1.7772	1.8727
RMSE	4.2372	3.7601	4.1412	4.1865	4.5209	4.5866	4.7961	5.1375
NAE	0.3914	0.3658	0.3878	0.3918	0.4181	0.4219	0.4277	0.4506
Real Mean Value	4.3851	4.1917	4.2707	4.3155	4.3783	4.4758	4.5211	4.6261

complete data. *Computers in Biology and Medicine* 36, 5 (2006), 516–525.

[16] Gonzalo Mateos and Georgios B Giannakis. 2013. Load curve data cleansing and imputation via sparsity and low rank. *IEEE Transactions on Smart Grid* 4, 4 (2013), 2347–2355.

[17] Khosrow Moslehi and Ranjit Kumar. 2010. A reliability perspective of the smart grid. *IEEE transactions on smart grid* 1, 1 (2010), 57–64.

[18] Jouni Peppanen, Xiaochen Zhang, Santiago Grijalva, and Matthew J Reno. 2016. Handling bad or missing smart meter data through advanced data imputation. In *2016 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 1–5.

[19] Seunghyoung Ryu, Minsoo Kim, and Hongseok Kim. 2020. Denoising Autoencoder-Based Missing Value Imputation for Smart Meters. *IEEE Access* 8 (2020), 40656–40666.

[20] Seunghyoung Ryu, Jaekoo Noh, and Hongseok Kim. 2017. Deep neural network based demand side short term load forecasting. *Energies* 10, 1 (2017), 3.

[21] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. 2020. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks* (2020).

[22] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. 2001. Missing value estimation methods for DNA microarrays. *Bioinformatics* 17, 6 (2001), 520–525.

[23] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*. 1096–1103.

[24] Long Wang, Zijun Zhang, Jia Xu, and Ruihua Liu. 2016. Wind turbine blade breakage monitoring with deep autoencoders. *IEEE Transactions on Smart Grid* 9, 4 (2016), 2824–2833.

[25] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic Flow Prediction via Spatial Temporal Graph Neural Network. In *Proceedings of The Web Conference 2020*. 1082–1092.

[26] Yang Weng, Rohit Negi, Christos Faloutsos, and Marija D Ilić. 2016. Robust data-driven state estimation for smart grid. *IEEE Transactions on Smart Grid* 8, 4 (2016), 1956–1967.

[27] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).