TG-SPRED: Temporal Graph for Sensorial Data PREDiction

ROUFAIDA LAIDI, Norwegian University of Science and Technology (NTNU), Trondheim, Norway
DJAMEL DJENOURI, University of the West of England, Bristol, UK
YOUCEF DJENOURI, University of South-Eastern Norway and NORCE Norwegian Research Centre, Norway
JERRY CHUN-WEI LIN, Western Norway University of Applied Sciences, Norway

This paper presents a novel approach to conserving sensor communication and sensing energy while extending the sensor network lifetime through sensor reading prediction. The proposed model generates readings, in each time slot, for a randomly selected subset of sensors that enter sleep mode. The high accuracy of the generated readings enables frequent and prolonged sleeping periods. Moreover, the proposed approach does not rely on any scheduling strategy, thus relaxing assumptions made in many existing works in the literature. Our focus is on event-based sensing, which poses more significant challenges and has received limited coverage in the current literature. Although events may appear independent, they often exhibit temporal and spatial correlations. Our proposed solution, TG-SPRED (Temporal Graph Sensor Prediction), captures these correlations and predicts the values of sleeping sensors. It leverages gated recurrent units (GRUs) to learn temporal features from sensing data and utilizes a graph convolutional network (GCN) to capture spatial features. The sensor network structure is represented as a graph, where weights are proportional to the distances between sensors. Furthermore, the temporal graph network is trained adversarially, employing a "critic" network that enhances the generation accuracy. This critic network uses the Wasserstein distance between real and generated data to estimate the performance of the generator. To evaluate the suggested approach, we employed four metrics and compared them with six state-of-the-art solutions: F-score, average energy consumption, average lifetime, and run time. The results demonstrate that the proposed method outperforms all other solutions regarding accuracy and energy savings.

CCS Concepts: • **Computer systems organization → Sensor networks**; **Embedded systems**; • **Hardware → Power and energy**.

Additional Key Words and Phrases: graph convolution neural network, adversarial training, sensor energy savings, spatiotemporal learning

ACM Reference Format:
Roufaida Laidi, Djamel Djenouri, Youcef Djenouri, and Jerry Chun-Wei Lin. 2022. TG-SPRED: Temporal Graph for Sensorial Data PREDiction. *ACM Trans. Sensor Netw.* 1, 1 (November 2022), 20 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Researchers, network designers, developers, and hardware manufacturers have all expressed concerns about increasing the network lifetime of smart devices in Internet of Things (IoT) applications. Various technologies such as Zigbee and Bluetooth Low Energy (BLE) for local area networks, and Lora & Lorawan and Sigfox for LPWAN (Low Power Wide Area Network) have addressed energy-efficient connectivity in IoT, catering to different ranges. Numerous research efforts have been dedicated to designing power-management policies and protocols, including routing protocols [12], medium access protocols [13], optimal relay node placement [11], data aggregation [4], broadcasting protocols [19], among others. Additionally, hardware-level proposals have emerged to incorporate wireless charging and energy harvesting capabilities into IoT devices [10].

However, these technologies may not fulfill the requirements of many applications. For instance, LoRa conserves energy through low-duty cycling and parameter adjustment [24], but this approach introduces latency, making it unsuitable for real-time applications, particularly in event-driven settings where data transmission occurs irregularly. This research work aims to maximize network lifetime by leveraging temporal and spatial features of sensory data while meeting the demands of IoT applications. The objective is to save sensor energy by predicting their values without compromising event detection accuracy. While many applications often require optimal sensing coverage, “continuous” monitoring of the entire field is rarely necessary, especially for event detection. Events typically occur rapidly and can be sensed simultaneously or in a time series by multiple sensors. If one sensor misses an event, others can detect it. Moreover, the positioning of sensors plays a role, as overlapping sensing fields can result in multiple sensors detecting the same event. By employing a model capable of learning spatial and temporal data correlations, the values of the sensors can be predicted, thereby reducing energy consumption in sensing and transmission.

Recently, accurate sensor reading prediction has gained attention in research. In the existing literature, two approaches have been proposed. The first approach involves predicting future values when all sensors are turned “off” based on their previous readings, leveraging temporal correlations among sensors with similar states simultaneously[23]. On the other hand, the second approach focuses on selectively activating a portion of the network and utilizing spatial correlations to infer the values of the sleeping sensors[9]. At the data collector level, which depending on the architecture, can be the cloud, a base station, or an edge device, we propose a model that generates the missing data of sleeping sensors. This model employs Generative Adversarial Networks (GANs), which have already demonstrated their effectiveness in data imputation tasks. GANs provide a robust framework for generating synthetic data that closely resemble the characteristics of the original sensor readings. By leveraging the capabilities of GANs, our model aims to accurately predict the values of the sleeping sensors, thereby completing the missing data and enabling comprehensive analysis and processing of the sensor network data. GANs work based on a game-theoretic situation where two deep learning models -the generator and the discriminator/critic - compete. The discriminator’s objective is to distinguish actual data from the generated data, while the generator strives to generate missing data samples that closely resemble the real data distribution. However, these models are limited to learning only similarities between data aspects, failing to comprehend the problem’s structure. To overcome this limitation, we propose utilizing deep graph neural networks (GNN) to model the generator. This technique extends deep learning to non-Euclidean domains, such as graphs, which characterize sensor networks. The generator is trained semi-supervised, with network visibility restricted to readings from active nodes. The output of the GNN is then passed to a gated recurrent unit (GRUs), a variant of recurrent neural networks, to capture temporal features.

TG-SPRED’s main contributions can be summarized as follows: 1) It conserves sensor communication and sensing energy in event-based applications by generating (predicting) the reading of sleeping sensors. The precision enables extended and frequent inactivity (sleep) periods and, thus, energy preservation. This is achieved through contributions 2-3 and confirmed in 4. 2) It learns temporal correlations in sensorial data and spatial structural information from a weighted undirected graph that reflects the coverage overlap between nodes. This is achieved by the generator modeled as a graph convolutional network (GCN) in GRU-based architecture. 3) It trains the generator against a critic network that learns to identify imputed data from actual data, thus speeding up training and enhancing accuracy. 4)The comparison with six state-of-the-art solutions using a real data set and four metrics confirms a clear improvement of the proposed model.

The remainder of the paper is presented as follows. Sec.2 presents the related work. Sec.3 explains the main components of the proposed framework. Sec.4 gives the experimental analysis part, where Sec.5 concludes the paper.

2 RELATED WORK

Low-Power Wide Area Network (LPWAN) solutions gained attention among industry and researchers due to their promise of boosting node lifetime in IoT networks. LoRa technology [28] promises kilometers of communication range, years of battery life, interference resistance, and many device simultaneous transmissions. However, Liando et al. [24] demonstrate that the claimed performance is contingent on fine-tuning the settings and that a long battery life necessitates a very low duty-cycling rate. Predicting sensory data is one intriguing solution, where machine learning-based approaches have been explored for generating sensor values. This section reviews works on predicting missing IoT values and solutions that explore spatiotemporal learning using graph convolution neural networks.

2.1 Prediction-based Solutions for Missing IoT Values

Diaz et al. [8] studied the possibility of forecasting data in sensors, and the results indicated that this approach reduces transmissions without compromising data quality. There are two types of prediction-based sensor energy-saving techniques: 1) “single prediction schemes” and 2) “dual prediction schemes.” One network node maintains the prediction model and generates sensor values in the first category. The central node and sensor nodes both contribute to the prediction process in the second category. The principal limit of the second category is the sensor nodes’ restricted capacity, where the energy used in calculations is likely to outweigh the energy saved by lowering transmissions.

Consequently, this paper targets solutions of the first category and focuses on single prediction schemes that allow energy savings in sensing and communication operations based on a central entity. Silvestri et al. [29] infer sensor readings using a Gaussian distribution. First, they presented strategies for selecting a collection of awake monitoring sensors. The values of the sleeping sensors are then deduced from the values of the awake sensors using the Gaussian joint distribution. Laidi et al. [23] described a new method that allows sensors to be turned off during periods when their data can be predicted, saving energy that would otherwise be used for sensing and transmission. For future predictions, the suggested method employs an extended short-term memory model that learns spatiotemporal patterns in sequences of sensory input. The sensors and the long short-term memory model collaboratively monitored the environment. They are guided by a reinforcement learning agent that makes dynamic decisions about whether to use the long short-term memory prediction or physical sensing to save energy while maintaining prediction accuracy. In the context of missing data imputation, Yoon et al. [34] investigated the application of a generative adversarial network at the data collector level for missing data imputation. However, their method does not learn from the structural information in data, making it inappropriate for learning spatial and temporal relationships. This may cause the generator’s training to be delayed and its generation accuracy to suffer. On the other hand, Spinelli et al. [30] presented data as a graph structure automatically generated by measuring the distance between data features. A GCN network processes the generated graph to generate missing data. However, the solution creates a graph node for each data entry, which results in a big graph for large datasets. Such graphs are impossible to process on data collectors with limited processing capabilities. Our work exploits node locations and sensing coverage to build the graph structure. Each graph node represents a sensor, which is computationally more optimal.

2.2 Spatiotemporal Graph Convolution Neural Network

Liu et al. [25] proposed the Social spatiotemporal Graph Convolutional Neural Network (Social-STGCNN), which models interactions as a graph and eliminates the requirement for aggregation methods. A kernel function was proposed to integrate the social interactions between pedestrians within the adjacency matrix. To replace the aggregation layers, the pedestrians' paths are modeled as a spatiotemporal graph. The graph edges represent the pedestrians' social interactions. A weighted adjacency matrix is also created, in which the kernel function quantitatively measures pedestrians' influence. Khodayar et al. [20] provided a graph deep learning model for learning powerful spatiotemporal features from wind speed and wind direction data in surrounding wind farms. An undirected network models the underlying wind farms, with each node representing a wind station. A long-short-term memory network extracts temporal information for each node. Inspired by the localized first-order approximation of spectral graph convolutions, a scalable graph convolutional deep learning architecture used the recovered temporal features to anticipate the wind-speed time series of the entire network nodes. At each wind station, the proposed network captures both spatial and deep temporal aspects of the wind data. Ali et al. [2] provided a graph deep learning model for learning powerful spatiotemporal features from wind speed and wind direction data in surrounding wind farms. An undirected network models the underlying wind farms, with each node representing a wind station. A long-short-term memory network extracts temporal information for each node. Inspired by the localized first-order approximation of spectral graph convolutions, a scalable graph convolutional deep learning architecture used the recovered temporal features to anticipate the wind-speed time series of the entire network nodes. Deng et al. [7] proposed a spatiotemporal graph convolutional adversarial network (STGAN). A spatiotemporal generator is created to anticipate normal traffic dynamics, and a spatiotemporal discriminator is created to determine whether an input sequence is real or fake. In both the spatial and temporal dimensions, there are strong correlations between surrounding data points. As a result, a new module that used the graph convolutional gated recurrent unit (GCGRU) is presented to assist the generator and discriminator in learning the spatiotemporal aspects of traffic dynamics and anomalies, respectively. The generator and discriminator can be employed as detectors independently after adversarial training, with the generator modeling regular traffic dynamics patterns and the discriminator providing detection criteria that vary with spatiotemporal variables. Wang et al. [31] investigated the novel topic of multivariate correlation-aware multiscale traffic flow prediction and proposed the MC-STGCN, a feature correlation-aware spatiotemporal graph convolutional network. In particular, given a road graph, a coarse-grained road graph is designed based on topology similarity and traffic flow similarity among the nodes. Then, a cross-scale spatial-temporal feature learning and fusion technique deals with fine and coarse-grained traffic data. A cross-scale graph convolution neural network is presented in the spatial domain to learn and fuse multi-scale spatial variables concurrently. A cross-scale temporal network of hierarchical attention is created to capture intra and inter-scale temporal correlations in the temporal domain.

2.3 Discussion

To our knowledge, no work in the literature considers the spatiotemporal information for predicting missing IoT values to save sensor energy. Motivated by the success of the hybrid combination of GCNs and recurrent neural networks in handling spatiotemporal data, we propose a hybrid GCN and GRU model to learn spatiotemporal dependencies between sensor readings. Besides, we train the model adversarially to endorse its data generation performance.

3 TG-SPRED FRAMEWORK

3.1 Overview and Problem Formulation

We suppose an IoT solution that comprises N event-based sensors (e.g., motion sensors). Following a star topology, the latter sends event data to a data collector (base station, edge, or cloud). In IoT architectures, the data collector commonly has higher computation and energy resources than the sensors, usually powered using cell batteries. The goal of the solution is to reduce the sensors' working time and, thus, the energy consumed for sensing and communication. At each time step, a part of the network is randomly turned off to save sensor energy. The ratio of sleeping sensors is a parameter that controls the number of sleeping sensors in each time slot. A higher value means more extended sleeping periods (homogeneously for all sensors), which reduces energy but causes more missing sensor readings. Therefore, we propose TG-SPRED model that runs on the data collector level and generates data that replaces the missing values. Our solution enhances real-time responsiveness in IoT environments by employing a central unit that directly predicts the values of dormant sensors, effectively bypassing the latency typically introduced by traditional duty cycling strategies. Our predictive model is powered by advanced algorithms capable of accurate forecasts, minimizing the risk of false negatives that could occur due to sensor inactivity. As a result, the integrity of real-time monitoring is preserved without compromising energy conservation goals often associated with sensor duty cycling. This predictive approach ensures a seamless data stream, avoiding the common issue of missing critical events during sensor inactivity periods. Consequently, our system delivers an uninterrupted monitoring experience, crucial for applications where immediate detection and action are imperative, such as security and safety monitoring in smart buildings. This method not only maintains continuous system vigilance but also aligns with energy efficiency objectives, striking a balance between operational immediacy and power management.

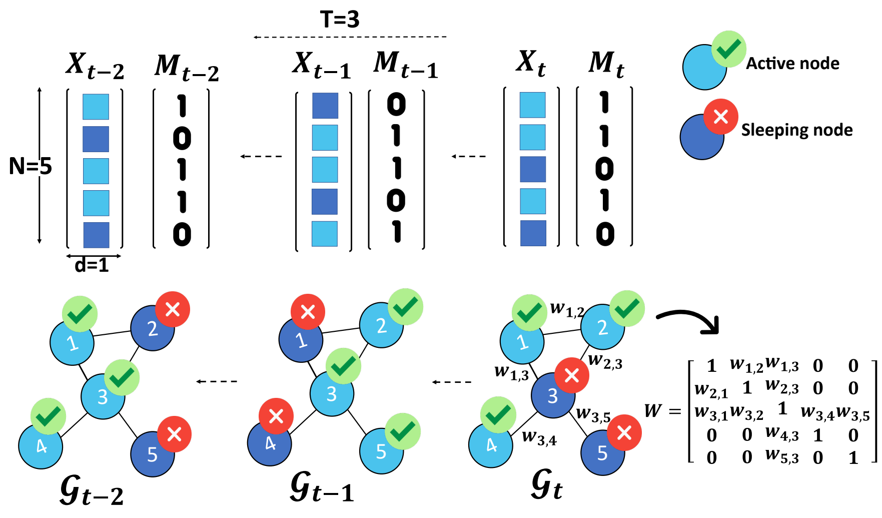
TG-SPRED comprises two neural networks: 1) a spatiotemporal generator and 2) an adversarial critic. The generator is a temporal graph network that learns spatial and temporal features in sensor readings and generates the sleeping sensor values. It combines a graph network that captures spatial features and a recurrent network that learns temporal ones. On the other hand, the critic's role is limited to the training phase by sending feedback about generation accuracy, which challenges the generator to enhance its performance.

The sensors are organized as a weighted graph with N nodes and an adjacency matrix $W \in \mathbb{R}^{N\times N}$. $X_t \in \{0,1\}^{N\times d}$ is the binary data for time step t collected by the sensor nodes, where 1 represents the presence of an event. d is the number of node attributes. It may represent multiple readings for a sensor at the time t, e.g., a node with multiple sensing boards, or the length of the historical time series, i.e., each time step is presented as a window of d readings. The binary data is either collected by binary sensors (e.g., motion sensors) or processed data from analog sensors, e.g., increasing temperature beyond a threshold can indicate a fire event. This paper makes abstraction of the signal processing phase, and data is deemed preprocessed and presented in a binary form. Finally, the overall network is presented as a sequence of T graphs, $\mathcal{G}_t = \langle X_t, W\rangle$, where T is the number of time sequences. The construction of the weighted adjacency matrix is discussed in Sec.3.2.

The missing data is captured by a binary mask, $M_t \in \{0,1\}^{N\times d}$, where each row m_t^i indicates the presence or absence of the reading in x_t^i: $m_t^{i,j} = 0$ if $x_t^{i,j}$ is missing and needs to be generated, the presence of the real sensor reading is indicated by $m_t^{i,j} = 1$. Fig.1 shows an example of the network for $N = 5$, $d = 1$, and $T = 3$.

Finally, the solution yields to generating $Y_t \in \{0,1\}^{N\times d}$, which represents the combination of the real and generated data as shown in Eq.(1):

$$Y_t = M_t \odot X_t + \overline{M_t} \odot \widetilde{Y}_t, \tag{1}$$



$$W = \begin{bmatrix} 1 & w_{1,2} & w_{1,3} & 0 & 0 \\ w_{2,1} & 1 & w_{2,3} & 0 & 0 \\ w_{3,1} & w_{3,2} & 1 & w_{3,4} & w_{3,5} \\ 0 & 0 & w_{4,3} & 1 & 0 \\ 0 & 0 & w_{5,3} & 0 & 1 \end{bmatrix}$$

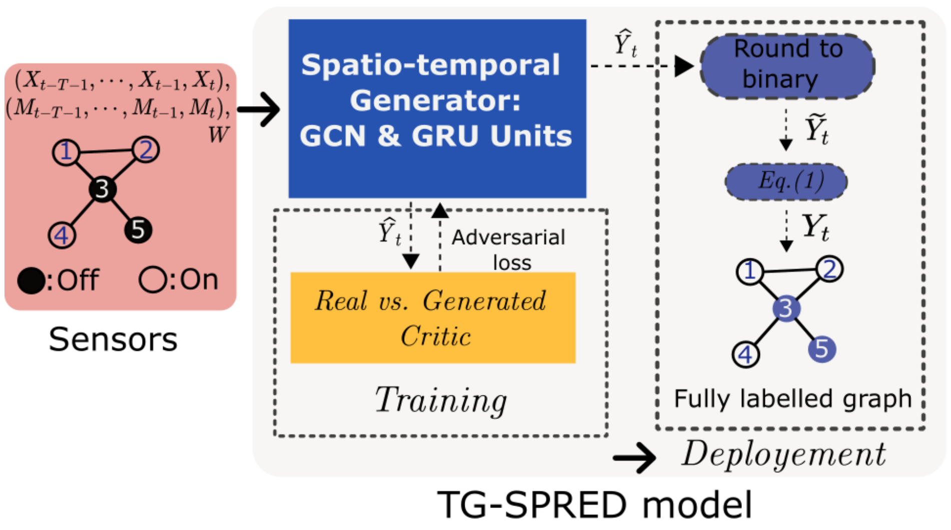
Fig. 1. Network representation.

where $\widetilde{Y}_t \in \{0,1\}^{N\times d}$ is the output of the TG-SPRED model, and $\odot$ is the Hadamard product. The goal is learning a mapping function f that learns spatial dependencies from W and temporal dependencies from T previous time sequences $X_t, X_{t-2}, \cdots, X_{t-T-1}$, as shown in Eq.(2). When deployed, the trained model uses data from the $T-1$ previously imputed time steps. The Architecture of the TG-SPRED network is detailed in Sec.3.3.

$$\widehat{Y}_t = f((X_{t-T-1}, \cdots, X_{t-1}, X_t), (M_{t-T-1}, \cdots, M_{t-1}, M_t), W) \tag{2}$$

3.2 Graph Construction

The sensor network is considered a weighted undirected graph of distributed interconnected sensors. Assume a graph $\mathcal{G}_t = \langle X_t, W\rangle$ represented by a pair, $(\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is a set of N vertices defining the network topology, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges representing the different connections between two different sensors. $(\mathcal{V}, \mathcal{E})$ are not variable over time, since sensor locations do not change. For each pair of sensors (i, j), we determine the coverage value used to build the edges. Thus, $(i, j) \in \mathcal{E}$ if the sensor j is covered by the sensing range of the sensor i, and vice-versa. We suppose all sensors have an equal sensing range; therefore, the graph is undirected. We define the weighted adjacency matrix $W \in \mathbb{R}^{N\times N}$ for the edge weights. Each value in W represents the coverage value of two sensors and is proportional to the P_{ij} probability that j covers i. It is given by: $W_{ij} = P_{ij} = e^{-\beta d_{ij}}$, where d_{ij} is the Euclidean distance between v_i and v_j, $\beta \in [0, 1]$ is the sensing capacity decay factor. It describes how fast the sensing decays with distance and depends on the sensor and the environment. Notice $W_{ij} = 0$ if there is no edge between i and j, and $W_{ii} = 1\ \forall i$.



$(X_{t-T-1}, \cdots, X_{t-1}, X_t)$,
$(M_{t-T-1}, \cdots, M_{t-1}, M_t)$,
W
●:Off ○:On
Sensors
Spatio-temporal Generator: GCN & GRU Units
$\hat{Y}_t$
Adversarial loss
Real vs. Generated Critic
Training
Round to binary
$\tilde{Y}_t$
Eq.(1)
Y_t
Fully labelled graph
Deployement
TG-SPRED model

Fig. 2. Overview of the TG-SPRED framework.

3.3 Spatiotemporal Sensor Data Generation

This section presents the proposed model, the Temporal Graph for Sensorial Data PREDiction (TG-SPRED), a spatiotemporal graph-based solution to elongate the lifetime of event-based IoT applications. Given a sequence of readings $X[t-T;t]$, the role of the model is to generate binary sensor readings at a time t to restore the missing readings of inactive sensors at this time step. The inactivity of the sensors is depicted by a mask $M[t-T;t]$. The solution combines temporal and spatial dimensions to approach the real data distribution. To accelerate and enhance the learning process, we introduce adversarial training. The process is summarized in Fig.3. The TG-SPRED model is presented herein, followed by the adversarial training.

3.3.1 *Temporal Graph Convolution Network.*

Spatial Dependencies. Classical deep learning algorithms fail to catch spatial dependencies in complex and non-Euclidean spaces like graphs. Therefore, Graph Neural Networks (GNN)[33] emerged as a class of methods that explicitly use the structural relationships between graph entities. Part of these techniques are Graph Convolutional Networks (GCN), which inspire the capacity of traditional Convolutional Neural Networks (CNN) to learn local spatial features in images and extend it to graphs. In this paper, we adopt the approach proposed by [22] to temporal and missing data. A traditional GCN layer propagation rule is defined as :

$$H^{(l+1)} = \phi(\tilde{D}^{-\frac{1}{2}}\tilde{W}\tilde{D}^{-\frac{1}{2}}H^{(l)}\omega^{(l)}), \tag{3}$$

where $\phi(.)$ is a non-linear activation function, $H^{(l)}$ is the output of layer l, and $\omega^{(l)}$ is a trainable weight matrix for l. $\tilde{W} = W + I_N$ is an adjacency matrix W with added self-loops using the identity matrix I_N. $\tilde{D}_{ii} = \sum_{j\in\mathcal{N}(i)} \tilde{W}_{ij}$ is the diagonal node degree. $\tilde{D}^{-\frac{1}{2}}\tilde{W}\tilde{D}^{-\frac{1}{2}}$ is a symmetric normalization of $\tilde{W}$. $\tilde{D}^{-\frac{1}{2}}\tilde{W}\tilde{D}^{-\frac{1}{2}}$ is calculated during the pre-processing.

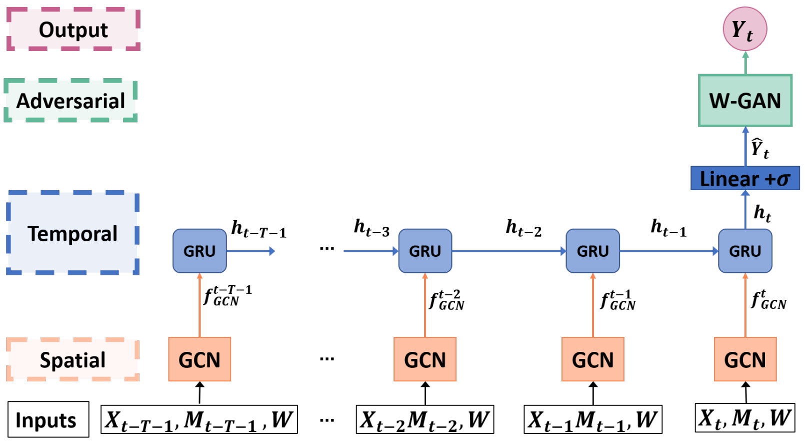


Fig. 3. TG-SPRED Network structure.

We define the GCN model as follows:

$$f_{GCN}^t(X_t, M_t, W) = \sigma(\widehat{W} ReLU(\widehat{W}(X_t \odot M_t)\omega_0)\omega_1), \tag{4}$$

where $f_{GCN}^t(X_t, M_t, W) \in \mathbb{R}^{N \times d}$ is the output of the model. X_t, W, and M_t are, respectively, features, adjacency, and mask matrices as defined in Sec. 3.1, and $\odot$ is the Hadamard product. $\widehat{W}$ and D_{ii} are calculated in the prepossessing as, $\widehat{W} = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, $D_{ii} = \sum_{j \in N(i)} W_{ij}$. Note that we do not add the identity matrix to W since the adjacency matrix already contains self-loops, as shown in Sec.3.2. $\omega_0 \in \mathbb{R}^{d \times H}$ is the first weight matrix (input to hidden), $\omega_1 \in \mathbb{R}^{H \times d}$ is the hidden to output weight matrix, d is the size of the input and output units and H the size of the hidden unit. Finally, ReLU(.) and $\sigma(.)$, are the REctified Linear Unit and sigmoid activation layer, respectively. The GCN model is repeated T times for each element in the time sequence.

Temporal Dependencies. We define a recurrent neural network (RNN) to catch temporal dependencies in the sequence of spatial embeddings generated by the GCN network. RNNs are the most used deep learning models for sequential data. However, because traditional RNNs suffer from gradient vanishing or explosion, they are limited to catching long-term dependencies. Therefore, variants such as LSTM[18] and GRU[6], which resolve this problem, are currently the most used. While LSTM and GRU have equal performances at multiple tasks, the structure of the GRU is less complex and, hence, faster to train; therefore, GRU is the model used in this work.

In our approach, the GRU cell receives as input f_{GCN}^t, i.e., the output of the GCN model at the time t, and the hidden state from the previous time step h_{t-1}. It then uses its reset, update, and new gates to calculate the new hidden state h_t. Eq. (5) shows the operations in different gates.

$$\begin{aligned} u_t &= \sigma(\omega_u[f_{GCN}^t(X_t, M_t, W), h_{t-1}] + b_u) \\ r_t &= \sigma(\omega_r[f_{GCN}^t(X_t, M_t, W), h_{t-1}] + b_r) \\ c_t &= tanh(\omega_c[f_{GCN}^t(X_t, M_t, W), (r_t \odot h_{t-1})] + b_c) \\ h_t &= u_t \odot h_{t-1} + (1 - u_t) \odot c_t \end{aligned} \tag{5}$$

where, $h_t \in \mathbb{R}^{N\times l}$, is the hidden state at the time t and the output of the model, f^t_{GCN} is the input at the time t, and the output of the GCN using Eq.4, $h_{(t-1)}$ represents the hidden state of the layer at the time $t-1$ (when $t > 1$) or the initial hidden state at the time 0, and r_t, u_t, c_t are the reset, update, and new gates, respectively. σ is the sigmoid function, $\odot$ is the Hadamard product.

Finally, the hidden state is passed to a linear layer and then to a sigmoid function, as shown in Eq.(6). $\omega_l \in \mathbb{R}^{l\times d}$ and $b_l \in \mathbb{R}^d$ are learnable weight matrix and bias vector, respectively. We use a sigmoid function in the output layer to generate the matrix $\widehat{Y}_t \in (0,1)^{N\times d}$, which is rounded to a binary matrix $\widetilde{Y}_t \in \{0,1\}^{N\times d}$.

$$\widehat{Y}_t = \sigma(h_t\omega_l + b_l) \tag{6}$$

Loss function. In this work, we train the model in a semi-supervised fashion, i.e., in each time slot, only the readings of active sensors are used as labels. Although it is possible to dedicate a training phase where the complete set of sensor readings is collected by keeping all the sensors active, the chosen approach reduces energy consumption. It extends the network's lifetime since collecting data for training may expand through several weeks. It also proves that the model approaches the real data distribution and is not overfitting.

Therefore, we compare the generated $\widehat{Y}_t$ to the values of active sensors in X_t, i.e., $m^{ij}_t = 1$. Since the generated data is binary, we use the binary cross-entropy loss. Hence, the loss function used to train the TG-SPRED model is presented as follows:

$$\mathcal{L} = -\frac{1}{T}\sum_{\tau=t-T-1}^{t}\sum_{i=1}^{N}\sum_{j=1}^{d}[x^{ij}_\tau m^{ij}_\tau \log \hat{y}^{ij}_\tau + (1 - x^{ij}_\tau m^{ij}_\tau)\log(1-\hat{y}^{ij}_\tau)] \tag{7}$$

3.3.2 *Adversarial Training with W-GAN.* The goal of the TG-SPRED framework is to use the spatial and temporal features to learn a function that approaches the distribution of sensor readings and, therefore, accurately generates the missing data. One family of neural networks that has proven its efficiency in approaching data distributions for generation is Generative Adversarial Networks (GAN)[14]. Following Wasserstein GAN (W-GAN) approach[3], we train a deep neural network called "critic" to provide feedback about the performance of the temporal-GCN. We then add the adversarial loss to the temporal-GCN loss (defined in Eq.(7)).

During the training, the critic learns a function c to compute the *Wasserstein* distance between the sensor readings of active sensors in X_t and the generated vector $\widehat{Y}_t$. We note these distributions p_r and p_g, respectively. The optimal function maximizes the distance between p_r and p_g. Thus, it increases the critic's capacity to differentiate real and generated data and challenges the temporal-GCN to approach X_t better. Accordingly, the W-GAN loss function is defined as:

$$\mathcal{L}_c = \sup_c \mathop{\mathbb{E}}_{x\sim p_r}[c(x)] - \mathop{\mathbb{E}}_{\tilde{x}\sim p_g}[c(\tilde{x})] \tag{8}$$

where x and $\tilde{x}$ are sampled from p_r and p_g, respectively. sup corresponds to the supremum (the least upper bound) of all possible functions. c a *1-Lipschitz* continuous function: everywhere continuously differentiable and its gradient's norm is at most 1 everywhere, i.e., c satisfies $|c(x) - c(y)| \le |x - y|, \forall x, \forall y$. c is learned during the training. To guarantee that c is a *1-Lipschitz* function, Gulrajani et al. [16] added a gradient penalty by calculating the gradient's norm for random samples to enforce the norm to be is at most 1 everywhere, $\hat{x}$. Implicitly, the Wasserstein loss with gradient penalty is calculated as:

$$\mathcal{L}_{cgp} = \underset{\tilde{x}\sim p_g}{\mathbb{E}}[c(\tilde{x})] - \underset{x\sim p_r}{\mathbb{E}}[c(x)] + \underbrace{\lambda \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}[(\|\nabla_{\hat{x}} c(\hat{x})\| - 1)^2]}_{\text{gradient's penalty}} \tag{9}$$

where $\hat{x} \sim p_{\hat{x}}$. $p_{\hat{x}}$ is sampled uniformly along straight lines between pairs of points sampled from p_g and p_r i.e., $\hat{x} = \epsilon x + (1-\epsilon)\tilde{x}$, where $\epsilon \sim U\,[0,1]$. λ is the gradient penalty coefficient, generally, takes the value 10[16].

While the critic is trained to minimize the loss in Eq.(9), the generator is trained to minimize $-\mathbb{E}_{\tilde{x}\sim p_g}[c(\tilde{x})]$. Eq.(10) shows the combination of the adversarial training loss with the temporal-GCN loss (Eq.(7)). We include the hyperparameter ζ to balance between the two losses.

$$\mathcal{L}' = \mathcal{L} - \zeta \underset{\tilde{x}\sim p_g}{\mathbb{E}}[c(\tilde{x})] \tag{10}$$

Algorithm 1 describes the training process of the TG-SPRED model. For every time step t in T, both the generator and critic are trained. As in [3, 16], the critic is trained $n_{critic} = 5$ times for each generator's iteration. We compute $\underset{x\sim p_r}{\mathbb{E}}[c(x)]$ and $\underset{\tilde{x}\sim p_g}{\mathbb{E}}[c(\tilde{x})]$ by sampling a random mini-batch x from active sensor readings, i.e., $X_t \odot M_t$, and $\tilde{x}$ its corresponding generated mini-batch from $\widehat{Y}_t$ (Eq.(6)). x and $\tilde{x}$ are fed to the critic, and the expectations are approximated using the critic's output averages. The critic weights θ are updated with $\mathcal{L}_{cgp}$ (Eq.(9)) and the generator weights W are updated using $\mathcal{L}'$ (Eq.(10)). For the gradient optimization of the two networks, we use the *Adam* optimization algorithm[21], which is more robust against the gradient vanishing problem.

4 PERFORMANCE EVALUATION

Context and Dataset. Our employed *MERLSense* dataset [27] consists of a meticulous collection of over 50 million motion sensor events from two floors of a workspace building, recorded with millisecond granularity over two years, noting the timestamp and sensor ID for each event. From this, we constructed a matrix where columns correspond to sensor IDs and rows represent the events within the same second, selecting and retaining readings from sensors of particular interest based on their floor map locations. This data represents a sequence of 3337087 consecutive seconds. To ensure a robust test bed, 20% of this dataset was set aside for testing. Our experimental setups evaluated our solution's scalability in 8, 16, and 32 node networks. Fig.4 displays the sensor placements and building layout. We based the interaction graph on a 3.5-meter sensing radius, aligning with the motion sensors' operational range during data collection. The experimental setup, including the dataset used and sensing parameters, is summarized in Table 1, providing a detailed overview of the key components of our study. The models were implemented using the PyTorch deep learning framework [26], with the PyTorch Geometric Temporal library [27] employed for the temporal generator, all trained on an *Nvidia GeForce RTX 2080 Ti* GPU.

Adversaries. TG-SPRED is compared against six state-of-the-art approaches: 1) GCN[1][22], 2) A3T-GCN [5] [2], 3) ASTGCN [17][3] 4) GAIN[4][34], 5) GINN[5][30], and 6) JGD[29]. GCN is a generic approach for deep learning on graphs, while A3T-GCN and ASTGCN combine GCN with GRU and attention mechanism. The comparison between TG-SPRED and GCN shows the influence of removing the adversarial and temporal training on the generator's performance. On

[1]https://github.com/tkipf/pygcn
[2]https://github.com/lehaifeng/T-GCN/tree/master/A3T-GCN
[3]https://github.com/guoshnBJTU/ASTGCN-r-pytorch
[4]https://github.com/jsyoon0823/GAIN
[5]https://github.com/spindro/GINN

Algorithm 1: The temporal GCN generator and critic training algorithm

```
Input: $n_{critic}$: the number of critic iterations per generator iteration; $s$: the batch size, $\lambda$: the critic's gradient
       penalty coefficient, $\alpha, \beta_1, \beta_2$: Adam hyperparameters.
Initialization critic parameters $\theta_0$ and generator parameters $W_0$;
while the training loss has not converged do
    for $t = 1$ to $T$ do
        for $j = 1$ to $n_{critic}$ do
            calculate $\widehat{Y}_t$; /* Eq.(6) */
            Sample a random batch $x \in \{0,1\}^{N \times s}$ from $X_t \odot M_t$ and its corresponding generated batch
              $\tilde{x} \in (0,1)^{N \times s}$ from $\widehat{Y}_t$;
            for $i = 1$ to $s$ do
                Sample a random number $\epsilon \sim U[0,1]$;
                $\hat{x} \leftarrow \epsilon x_{*,i} + (1-\epsilon)\tilde{x}_{*,i}$;
                $\mathcal{L}_{cgp}^{(i)} \leftarrow c_\theta(\tilde{x}_{*,i}) - c_\theta(x_{*,i}) + \lambda[(\|\nabla_{\hat{x}} c_\theta(\hat{x})\| - 1)^2]$; /* Eq.(9) */
            end
            /* Update the critic: */
            $\theta \leftarrow Adam(\nabla_\theta \frac{1}{s}\sum_{i=1}^{s} \mathcal{L}_{cgp}^{(i)}, \theta, \alpha, \beta_1, \beta_2)$;
        end
        /* Update the generator: */
        Sample a random batch $x \in \{0,1\}^{N \times s}$ from $X_t \odot M_t$ and its corresponding generated batch $\tilde{x} \in (0,1)^{N \times s}$
          from $\widehat{Y}_t$ ;
        $W \leftarrow Adam(\nabla_W \mathcal{L} - \zeta \frac{1}{s}\sum_{i=1}^{s} c_\theta(\tilde{x}_{*,i}), W, \alpha, \beta_1, \beta_2)$; /* Eq.(10) */
    end
end
```

Table 1. Overview of Dataset and Sensing Parameters.

Parameter	Value
Dataset	*MERLSense* dataset [32]
Dataset Size	Total:3337087, Test:20%
Sensor Type	Motion sensor for occupancy detection
Event Type	Binary motion detection
Sensing Radius	3.5 meter
Network Size	8, 16, 32

the other hand, comparing with A3T-GCN and ASTGCN enables examining the benefits of adversarial training and the proposed GRU architecture. Furthermore, the comparison with GAIN demonstrates the impact of using adversarial training for missing data generation without including spatial and temporal feature learning. GINN used both a GCN network and adversarial training. Nevertheless, each node in the graph represents a data input. Such a presentation is not optimal for sensor data. The latter is continuously generated at a high rate. Furthermore, it does not include temporal feature learning. GAIN and GINN target missing data in general. However, their tests are mainly conducted on sensor data. Finally, JGD shares with the current solution the goal of generating the data of the passive sensors to preserve their energy. The authors propose an active node selection strategy and present sensor data as a Gaussian distribution.

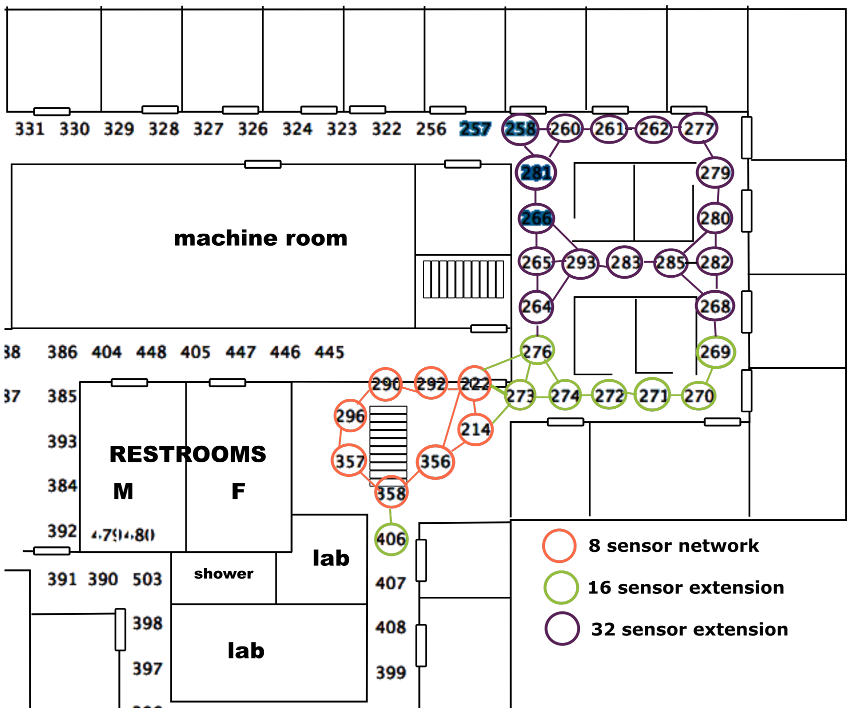


Fig. 4. The sensor graph and the building floorplan.

Evaluation Metrics. We compare the solution using four metrics. First, we measured the F-score[15] while varying the percentage of missing data. F-score is a harmonic mean of recall and precision, two metrics that estimate the accuracy of binary classification. The precision is the sum of correctly generated events (true positives) divided by the number of events in the generated data, including false positives. The recall is the number of correctly generated events divided by the number of all the events that should have been detected. The second and third metrics are the average energy consumed and the average lifetime, respectively, for a sensor. Both the average energy and lifetime are measured while varying the F-score. Finally, we tested the impact of extending the size of the network. Three network sizes were tested while measuring the run times and the previously mentioned metrics for the compared solutions.

4.1 Model Training and Hyperparameters Choice

We conceived a graph of $N = 8$ sensors. To decide the optimal value for T, we calculated the F-score for training and test sets while varying T. The results are depicted in Fig.5. The results show that starting $T = 1000$, the F-score stops increasing. Hence, we choose this value for T, which is computationally efficient. We fixed d to 3600 second, representing a time window of one hour with one-second granularity. The increase of the F-score for the first values also demonstrates the benefit gained from including the temporal dependencies in estimating sensor readings.

The TG-SPRED model is trained using the Adam optimizer for both the temporal GCN and the critic. The generator's learning rate is 10^{-4}, and the critic's learning rate is 5.10^{-5}. Tab.2 shows the selected hyperparameter values.

Model	Hyperparameters	Value
GCN Generator	d	3600 second
	Epoch	50
	Activation	ReLu
	Learning Rate	5.10^{-4}
	Weight Decay	0.1
	β_1, β_2	$(0.9, 0.999)$
Critic	Learning Rate	5.10^{-5}
	Critic Iteration	5
	Embedding dimension	512
	λ Gradient penalty	10
	β_1, β_2	$(0.9, 0.999)$

Table 2. Selected hyperparameters values.

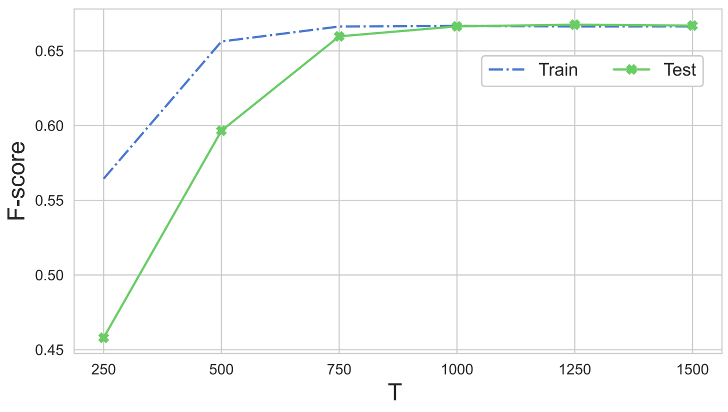


Fig. 5. F-score per T for 50% missing data.

Fig.6 shows the loss functions $\mathcal{L}$(Eq.(7)), $\mathcal{L}_{cgp}$(Eq.(9)), and $\mathcal{L}'$ (Eq.(10)) through the epochs with a missing data percentage fixed to 50% for both the train and test sets. We can notice that the loss for the generator decreases, which implies that the generators learned to approach the real dataset. In contrast, the critic's loss increases, indicating that the critic learned to differentiate between the generated and real data. Fig.7 indicated the F-score per epoch for 50% missing data. We notice that the F-score for test and train sets becomes close after 35 epochs, which indicates that the model is generalizing and not overfitting the data.

4.2 Generation Accuracy

The F-score in a network of 8 sensors is presented in Fig. 8, which shows that the performance of TG-SPRED is better than that of the compared solutions. Compared to GIIN, the outcomes confirm the effectiveness of the suggested formulation for the weighted graph that considers the sensor locations and their sensing range.

Outperforming A3T-GCN was due to the integration of adversarial training. A3T-GCN is considerably superior to regular GCN, which shows that temporal feature learning is primordial and spatial learning is not exclusively efficient.

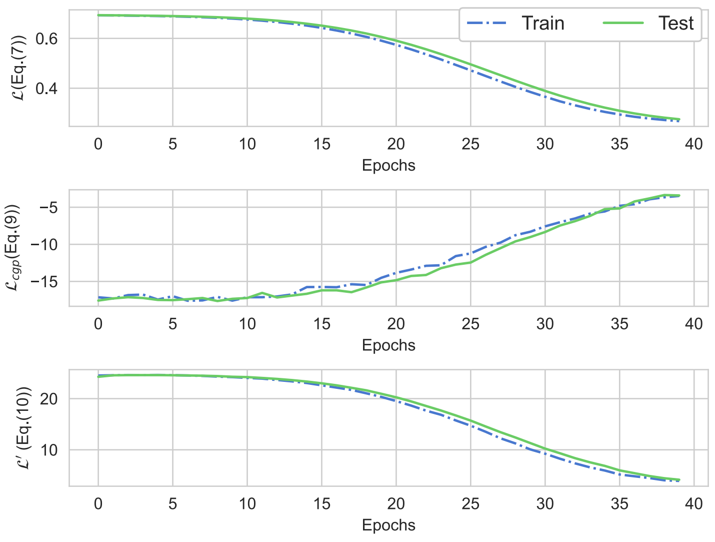


Fig. 6. $\mathcal{L}$, $\mathcal{L}_{cgp}$, and $\mathcal{L}'$ loss functions per epoch for 50% missing data.

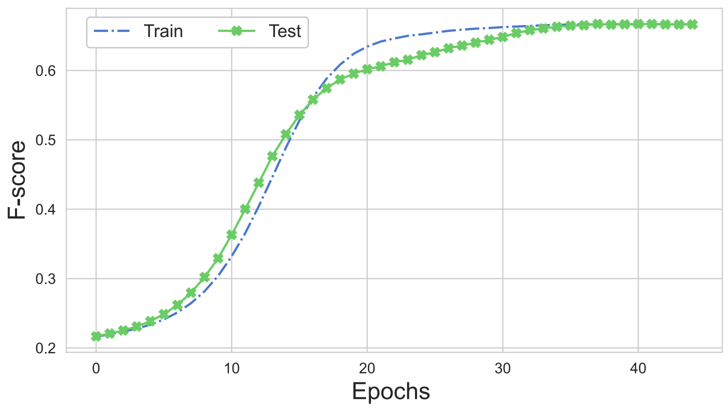


Fig. 7. F-score per epoch for 50% missing data

Despite ASTGCN and A3T-GCN utilizing both spatiotemporal learning and attention mechanisms, ASTGCN's performance is notably poorer than A3T-GCN. This is due to the solution structure, which necessitates high computational resources. As we use a time window of 3600 seconds (instead of 24 used in the original paper), we must lower other hyperparameter values during training, which impacts the solution's performance. In conclusion, computationally effective models are required when dealing with applications that necessitate high temporal granularity.

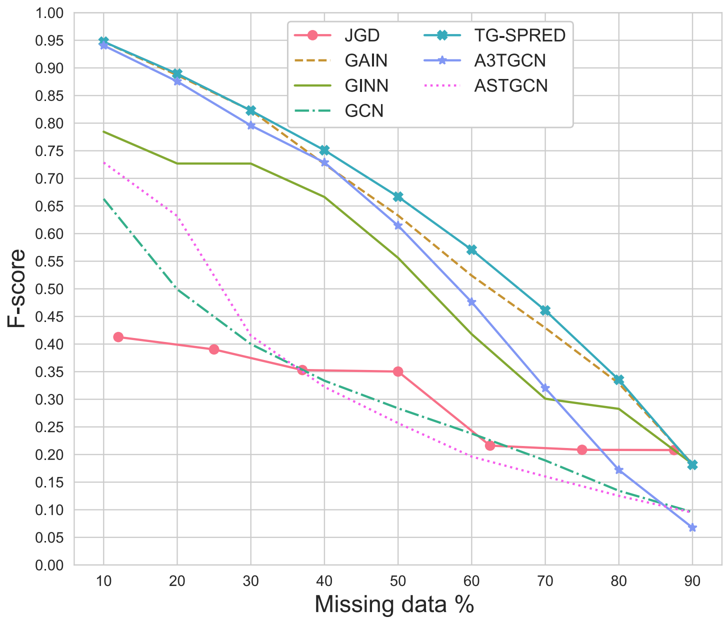


Fig. 8. F-score for 8 sensors per missing data %

The performances of TG-SPRED and GAIN are fairly comparable when the missing data is less than 30% or more than 80%, indicating that the two systems learn equally when a high or low quantity of data is provided. However, TG-SPRED is superior between 30% and 80%. All solutions have a poor F-score with 90% or more missing data, which highlights how challenging it is to train models with a small amount of data. A more realistic scenario has typical missing data percentages within a range that allows accuracy preservation and energy efficiency. The findings demonstrate that TG-SPRED greatly outperforms GAIN and all other solutions for these values. Additionally, JGD exhibits reduced variation in performance, suggesting that Gaussian distributions cannot adequately describe event-based sensing data.

4.3 Energy Conservation Performance

4.3.1 Energy Consumption. We measured the average energy in *milliwatt second* (mWs) consumed by one sensor while varying the F-score. By taking a commercialized wireless motion detector as a reference[1], the following energy model has been considered: the sensor is powered with a *CR2032* coin cell battery of $3V$ and 240 mAh. It consumes $e_{act} = 1.57$ mA in active mode for 56.66 ms, $e_{stb} = 3.45 .10^{-3}$ mA in standby mode, and $e_{inact} = 2.16 .10^{-3}$ mA in shutdown.

The energy in mWs equals the product of the Power in mW and Time in s, and the Power equals Voltage in V times Current in mA, i.e., $E = V \times I \times t$. Eq.(11) shows the calculation of the consumed energy by sensor i at time slot t. The time slot duration equals one second. Y_t and M_t are the model's output and mask matrix as defined in Sec.3.1. Finally, the average energy in mWs consumed by one sensor over T time slots is: $E = \frac{1}{N} \sum_{i=1}^{N} \sum_{\tau=t-T-1}^{t} E_{\tau}^{i}$, where N is the

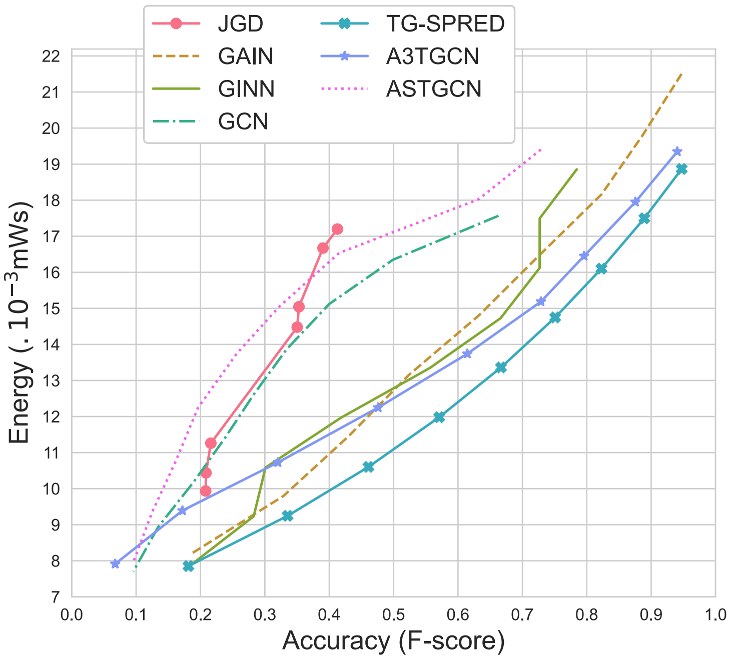


Fig. 9. Average energy in mWs consumed by one sensor per F-score

total number of sensors and T is the number of time slots.

$$E_t^i = 3 \times [(e_{act} \times 56.66\ .10^{-3} + e_{stb} \times 0.94334) \underbrace{\sum_{j=1}^{d} y_t^{ij} m_t^{ij}}_{\text{events}} + e_{stb} \underbrace{\left(\sum_{j=1}^{d} m_t^{ij} - \sum_{j=1}^{d} y_t^{ij} m_t^{ij}\right)}_{\text{stand by}} + e_{inact} \underbrace{(d - \sum_{j=1}^{d} m_t^{ij})}_{\text{inactive}}] \tag{11}$$

Fig. 9 depicts the average energy in mWs consumed by one sensor while varying the F-score. The figures demonstrate that TG-SPRED delivers the best accuracy/energy compromise.

4.3.2 *Sensor Lifetime* .

The same energy model was used to measure the battery's lifetime using Eq.(12).

$$\text{Battery Life} = \frac{\text{Battery Capacity (mAh)}}{\text{I (mA)}} \times \text{Derating factor,} \tag{12}$$

where the derating factor depends on external factors that affect batteries' lifetime, and I is the average Current of one sensor calculated by: $I = E/(V * T)$, where E, V, and T are average energy, voltage, and time as defined in Sec.4.3.

Fig. 10 shows the average lifetime for one sensor per F-score. In line with the results from Fig. 9, TG-SPRED guarantees the longest lifetime of all the compared solutions.

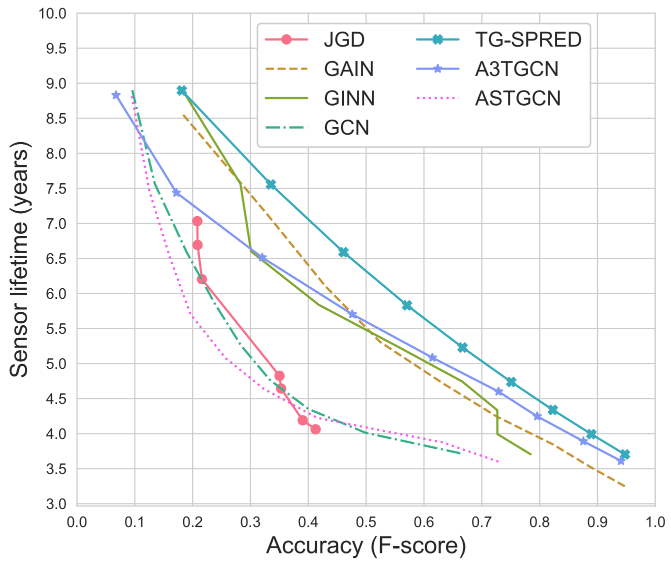
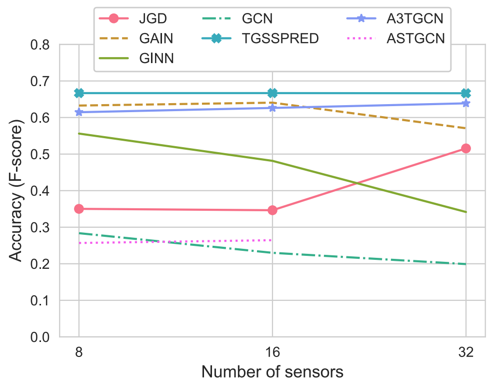


Fig. 10. Average lifetime for one sensor per F-score

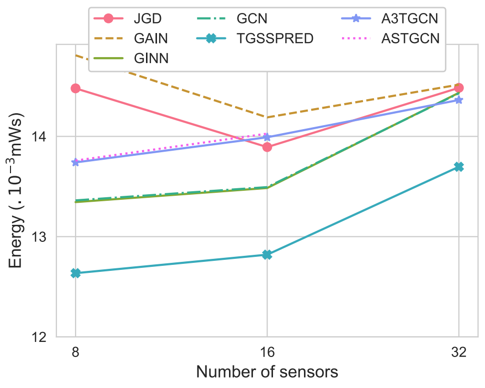
4.4 Extending the Network Size

In this part, the performance of the solutions is examined in different network sizes, 8, 16, and 32 sensors, while setting the missing data percentage to 50%. Given the star sensor network architecture and a 10m communication range, the network's maximum size was restricted to 32 sensors. A higher value in the number of sensors would reflect an unrealistic communication scenario for the dataset used in the experiment. Because ASTCGN necessitates high computational resources, we couldn't extend the network size to 32. As shown in Fig.11a, the F-score of TG-SPRED is the highest compared to all solutions and is unaffected by the number of sensors. The F-score for TG-SPRED, A3tGCN, and ASTCGN is stable and unaffected by the number of nodes, while JGD has a better F-score for 32 sensors. Conversely, GAIN, GIIN, and GCN performance decreases with the increase in the number of sensors. Fig.11b and Fig.11c display the average energy consumed by one sensor and its lifetime. While TG-SPRED demonstrates the lowest energy consumption, this metric decreases with the number of sensors, like most solutions. Similar results are shown for network lifetime, with TG-SPRED having the highest values. However, the lifetime drops when extending the network.

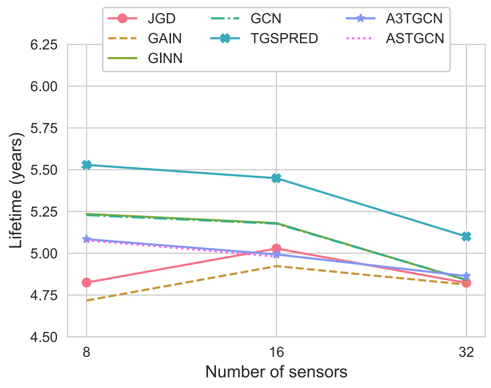
Fig. 11d plots runtime, i.e., the time needed for running the solution using a 2.80 GHz Intel Core $i7-7700HQ$ processor with 16GB $DDR4$ RAM. ASTCGN couldn't run under this hardware configuration. Hence, it is not included in these tests. Due to its data structure and attention mechanism, A3tGCN has the highest runtime, followed by GINN, whose network topology necessitates the creation of a node for each data entry, leading to a large graph and adjacency matrix. TG-SPRED, GCN, and JGD runtimes are comparable and almost identical, with JGD being slightly faster. However, as shown earlier, TG-SPRED significantly improves the other metrics, and TG-SPRED exhibits continuous runtime, confirming its scalability.



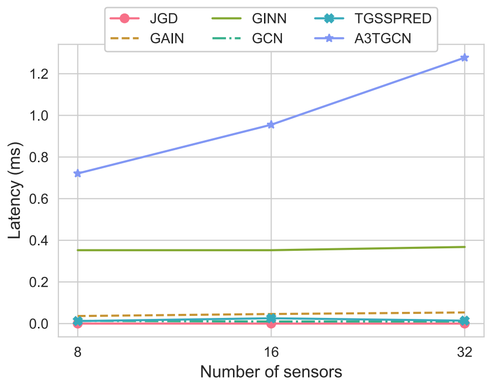
(a) F-score



(b) Average energy for one sensor



(c) Average lifetime for one sensor



(d) Computational run time

Fig. 11. Varying network size for 50% of missing data

5 CONCLUSION AND FUTURE WORK

The problem of predicting sensor readings to preserve energy in event-based IoT applications has been considered in this work, and a new solution has been proposed. It enables deactivating part of the sensors and synthesizing the missing data by leveraging the network's temporal correlation information, network structural information, and readings from the remaining active sensors. The proposed approach makes the abstraction of sensors' activity cycles and uses a generic random scheme. This makes it more general, as it does not depend on any duty-cycling scheduling method. The missing data is generated by an adversarially trained hybrid model combining GRU and GCN. We tested the approach on a real dataset and compared it to six other solutions from the literature using four metrics. The results revealed that the proposed method provides optimal accuracy and energy savings balance.

This solution is a continuation of our previous work [23] in which we utilized an LSTM to detect temporal connections in sensor readings and a reinforcement learning agent to decide when switching all the sensors on or off optimally. The current work considers the sensor network's structural information, and advanced machine learning models have been accordingly used. Future work targets extending further the size of the network for higher scalability. This will result in many disjoint subgraphs that traditional Deep Graph Networks do not process. A method of global representation for these disjoint graphs will be needed.

6 ACKNOWLEDGMENTS

This work was supported by the authors' institutions, as well as the Arab–German Young Academy of Sciences and Humanities (AGYA) through the German Federal Ministry of Education and Research (BMBF) under Grant 01DL20003

REFERENCES

[1] [n. d.]. Ultra-Low-Power Wireless PIR Motion Detector for Cost-Optimized Systems Reference Design. https://www.ti.com/lit/ug/tiducu5/tiducu5.pdf. Accessed: 2019-08-06.

[2] Ahmad Ali, Yanmin Zhu, and Muhammad Zakarya. 2022. Exploiting dynamic spatio-temporal graph convolutional neural networks for citywide traffic flows prediction. *Neural networks* 145 (2022), 233–247.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* (2017).

[4] Miloud Bagaa, Mohamed F. Younis, Djamel Djenouri, Abdelouahid Derhab, and Nadjib Badache. 2015. Distributed Low-Latency Data Aggregation Scheduling in Wireless Sensor Networks. *ACM Trans. Sens. Networks* 11, 3 (2015), 49:1–49:36.

[5] Jiandong Bai, Jiawei Zhu, Yujiao Song, Ling Zhao, Zhixiang Hou, Ronghua Du, and Haifeng Li. 2021. A3t-gcn: Attention temporal graph convolutional network for traffic forecasting. *ISPRS International Journal of Geo-Information* 10, 7 (2021), 485.

[6] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[7] Leyan Deng, Defu Lian, Zhenya Huang, and Enhong Chen. 2022. Graph Convolutional Adversarial Networks for Spatiotemporal Anomaly Detection. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[8] Gabriel Martins Dias, Boris Bellalta, and Simon Oechsner. 2016. On the importance and feasibility of forecasting data in sensors. *CoRR* abs/1604.01275 (2016). arXiv:1604.01275 http://arxiv.org/abs/1604.01275

[9] Djamel Djenour, Roufaida Laidi, and Youcef Djenouri. 2022. Deep Learning for Estimating Sleeping Sensor's Values in Sustainable IoT Applications. In *2022 International Balkan Conference on Communications and Networking (BalkanCom)*. 147–151. https://doi.org/10.1109/BalkanCom55633.2022.9900817

[10] Djamel Djenouri and Miloud Bagaa. 2015. Energy harvesting aware relay node addition for power-efficient coverage in wireless sensor networks. In *2015 IEEE International Conference on Communications, ICC 2015, London, United Kingdom, June 8-12, 2015*. IEEE, 86–91. https://doi.org/10.1109/ICC.2015.7248303

[11] Djamel Djenouri and Miloud Bagaa. 2017. Energy-Aware Constrained Relay Node Deployment for Sustainable Wireless Sensor Networks. *IEEE Trans. Sustain. Comput.* 2, 1 (2017), 30–42.

[12] Djamel Djenouri and Ilangko Balasingham. 2011. Traffic-Differentiation-Based Modular QoS Localized Routing for Wireless Sensor Networks. *IEEE Trans. Mob. Comput.* 10, 6 (2011), 797–809. https://doi.org/10.1109/TMC.2010.212

[13] Messaoud Doudou, Djamel Djenouri, José M. Barceló-Ordinas, and Nadjib Badache. 2016. Delay-efficient MAC protocol with traffic differentiation and run-time parameter adaptation for energy-constrained wireless sensor networks. *Wirel. Networks* 22, 2 (2016), 467–490.

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. arXiv:1406.2661 [stat.ML]

[15] Cyril Goutte and Eric Gaussier. 2005. A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. In *Advances in Information Retrieval*, David E. Losada and Juan M. Fernández-Luna (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 345–359.

[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in neural information processing systems*. 5767–5777.

[17] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 922–929.

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[19] Mustapha Khiati and Djamel Djenouri. 2015. BOD-LEACH: broadcasting over duty-cycled radio using LEACH clustering for delay/power efficient dissimilation in wireless sensor networks. *Int. J. Commun. Syst.* 28, 2 (2015), 296–308. https://doi.org/10.1002/dac.2669

[20] Mahdi Khodayar and Jianhui Wang. 2018. Spatio-temporal graph deep neural network for short-term wind speed forecasting. *IEEE Transactions on Sustainable Energy* 10, 2 (2018), 670–681.

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[22] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[23] Roufaida Laidi, Djamel Djenouri, and Ilangko Balasingham. 2021. On Predicting Sensor Readings With Sequence Modeling and Reinforcement Learning for Energy-Efficient IoT Applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2021).

[24] Jansen C Liando, Amalinda Gamage, Agustinus W Tengourtius, and Mo Li. 2019. Known and unknown facts of lora: Experiences from a large-scale measurement study. *ACM Transactions on Sensor Networks (TOSN)* 15, 2 (2019), 1–35.

[25] Ryan Wen Liu, Maohan Liang, Jiangtian Nie, Yanli Yuan, Zehui Xiong, Han Yu, and Nadra Guizani. 2022. STMGCN: Mobile Edge Computing-Empowered Vessel Trajectory Prediction Using Spatio-Temporal Multi-Graph Convolutional Network. *IEEE Transactions on Industrial Informatics* (2022).

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic Differentiation in PyTorch. In *NIPS Autodiff Workshop*.

[27] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzman Lopez, Nicolas Collignon, and Rik Sarkar. 2021. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 4564–4573.

[28] Olivier BA SELLER and Nicolas Sornin. 2016. Low power long range transmitter. US Patent 9,252,834.

[29] Simone Silvestri, Rahul Urgaonkar, Murtaza Zafer, and Bong Jun Ko. 2018. A Framework for the Inference of Sensing Measurements Based on Correlation. *ACM Trans. Sen. Netw.* 15, 1, Article 4 (Dec. 2018), 28 pages. https://doi.org/10.1145/3272035

[30] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. 2020. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks* 129 (2020), 249–260. https://doi.org/10.1016/j.neunet.2020.06.005

[31] Senzhang Wang, Meiyue Zhang, Hao Miao, Zhaohui Peng, and Philip S Yu. 2022. Multivariate Correlation-aware Spatio-temporal Graph Convolutional Networks for Multi-scale Traffic Prediction. *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, 3 (2022), 1–22.

[32] C.R. Wren, Y.A. Ivanov, D. Leigh, and J. Westhues. 2007. The MERL Motion Detector Dataset. In *Workshop on Massive Datasets (MD)*. 10–14. https://www.merl.com/publications/TR2007-069

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[34] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GAIN: Missing Data Imputation using Generative Adversarial Nets. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 5689–5698.