PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
الجمهـوريـــة الجـزائـريـــة الديـمـقـراطيـــة الشـعـبيـــة
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
وزارة التعليـم العالـي والبحـث العلمـي
HIGHER SCHOOL OF COMPUTER SCIENCE AND DIGITAL TECHNOLOGIES-BEJAIA
المـدرسة العـليا في علـوم وتكنـولوجيـات الاعـلام الالـي بجـاية

Dissertation Submitted to the Department Of Computer Science in Partial Fulfillment of the Requirements for Master's Degree in Computer Science

Specialty: Artificial Intelligence and Data Sciences

Submitted By:

Ms. *BOUTAOUI Nada*

Literature Review On Energy Efficiency In Wireless Sensor Networks

Supervised by:

Pr. *BAGAA Miloud - UQTR*

Pr. *AZOUAOU Faiçal - ESTIN*

Members of jury:

▪ **Dr.** AMROUNI Samia	President	ESTIN
▪ **Mr.** Yanis HAMMOUM	Examiner	ESTIN
▪ **Dr.** KACI Amina	Examiner	ESTIN
▪ **Dr.** CHELOUAH Leila	Examiner	ESTIN

Academic year: 2023/2024

Abstract

Energy efficiency in WSNs is a problem of resource optimization among sensor nodes that needs to meet the operational constraints for maximum network lifetime and performance. In essence, achieving optimal energy efficiency in WSNs means bing capable of balancing power consumptions, quality of data, and reliability of networks within various deployment scenarios. This issue has become increasingly complicated due to the rapid growth of IoT applications, heterogeneous sensor technologies, and dynamic operation environments.

Because of this, it has become a topic of top priority for researchers who are working on the development of proficient methods so that energy management can be done easily and effectively by WSNs operators and designers.

The present research work includes a bibliographic study on different approaches followed in WSN energy optimization, focusing on the most recent advances that have incorporated ML and Graph-based techniques. In the following, we will talk about some classic energy-saving approaches, modern ML techniques including Reinforcement Learning (RL), and the potential of Graph Convolutional Networks (GCNs) and Federated Learning (FL) in addressing energy efficiency challenges in WSNs.

Keywords: Wireless Sensor Networks, Energy Efficiency, Machine Learning, Graph Convolutional Networks, Federated Learning, Reinforcement Learning.

ملخص

يُعتبر تحسين كفاءة الطاقة في شبكات الاستشعار اللاسلكية (WSNs) تحدياً كبيراً يتعلق بإدارة الموارد بين العقد الحسية لضمان تحقيق عمر أطول للشبكة وأداء عالي. في الأساس، الوصول إلى كفاءة الطاقة المثلى يتطلب تحقيق توازن بين استهلاك الطاقة، جودة البيانات، وموثوقية الشبكة في مختلف بيئات التشغيل. هذه المشكلة أصبحت أكثر تعقيداً بسبب النمو السريع في تطبيقات إنترنت الأشياء، وتنوع تقنيات الاستشعار، والتغيرات الديناميكية في البيئات التشغيلية.

نتيجة لذلك، أصبحت كفاءة الطاقة موضوعاً ذو أولوية للباحثين الذين يعملون على تطوير أساليب فعّالة، لتمكين مشغلي ومصممي الشبكات من إدارة الطاقة بسهولة وكفاءة.

يتناول هذا البحث مراجعة شاملة للأساليب المختلفة المستخدمة لتحسين كفاءة الطاقة في شبكات الاستشعار اللاسلكية، مع التركيز على أحدث الابتكارات التي تشمل تقنيات تعلم الآلة والطرق المعتمدة على الرسوم البيانية. سنستعرض بعض الأساليب الكلاسيكية لتوفير الطاقة، بالإضافة إلى التقنيات الحديثة مثل التعلم المعزز ،(RL) ودور شبكات الالتفاف البياني (GCNs) والتعلم الفيدرالي (FL) في معالجة تحديات كفاءة الطاقة في هذه الشبكات.

الكلمات المفتاحية: شبكات الاستشعار اللاسلكية، كفاءة الطاقة، تعلم الآلة، شبكات الالتفاف البياني، التعلم الفيدرالي، التعلم المعزز.

Résumé

L'efficacité énergétique dans les réseaux de capteurs sans fil (WSN) est un problème d'optimisation des ressources entre les nœuds capteurs, qui doit respecter les contraintes opérationnelles pour maximiser la durée de vie et les performances du réseau. En substance, atteindre une efficacité énergétique optimale dans les WSN signifie être capable d'équilibrer la consommation d'énergie, la qualité des données et la fiabilité des réseaux dans différents scénarios de déploiement. Ce problème est devenu de plus en plus complexe en raison de la croissance rapide des applications IoT, des technologies de capteurs hétérogènes et des environnements opérationnels dynamiques.

Pour cette raison, il est devenu un sujet de grande priorité pour les chercheurs qui travaillent au développement de méthodes efficaces afin que la gestion de l'énergie puisse être réalisée facilement et efficacement par les opérateurs et les concepteurs de WSN.

Le présent travail de recherche comprend une étude bibliographique sur différentes approches suivies dans l'optimisation énergétique des WSN, en se concentrant sur les avancées les plus récentes qui ont intégré des techniques d'apprentissage automatique et des méthodes basées sur les graphes. Dans ce qui suit, nous parlerons de certaines approches classiques d'économie d'énergie, des techniques modernes d'apprentissage automatique, y compris l'apprentissage par renforcement (RL), et du potentiel des réseaux de neurones convolutifs sur graphes (GCN) et de l'apprentissage fédéré (FL) pour relever les défis de l'efficacité énergétique dans les WSN.

Mots-clés : Réseaux de capteurs sans fil, efficacité énergétique, apprentissage automatique, réseaux convolutifs sur graphes, apprentissage fédéré, apprentissage par renforcement.

Abbreviations

AWGN: **A**dditive **W**hite **G**aussian **N**oise

CNN: **C**onvolutional **N**eural **N**etwork

CRPS: **C**ontinuous **R**anked **P**robability **S**core

DPM: **D**iffusion **P**robabilistic **M**odel

DP: **D**ifferential **P**rivacy

DRL: **D**eep **R**einforcement **L**earning

DQN: **D**eep **Q**-**N**etworks

EC: **E**vent **C**orrelations

FedAvg: **F**ederated **A**veraging

FedDyn: **F**ederated **D**ynamic **A**veraging

FedProx: **F**ederated **P**roximal

FL: **F**ederated **L**earning

GAN: **G**enerative **A**dversarial **N**etwork

GAT: **G**raph **A**ttention **N**etwork

GCN: **G**raph **C**onvolutional **N**etwork

GNN: **G**raph **N**eural **N**etwork

GSAVES: **G**raph **S**ensor **A**d**V**ersarial for **E**nergy **S**aving

HMM: **H**idden **M**arkov **M**odel

JGD: **J**oint **G**aussian **D**istribution

LDP: **L**ocal **D**ifferential **P**rivacy

LSTM: **L**ong **S**hort-**T**erm **M**emory

MAC: **M**edia **A**ccess **C**ontrol

MAE: **M**ean **A**bsolute **E**rror

MARL: **M**ulti-agent **R**einforcement **L**earning

MDP: **M**arkov **D**ecision **P**rocess

ML: **M**achine **L**earning

NN: **N**eural **N**etwork

PCA: **P**rincipal **C**omponent **A**nalysis

ReLU: **R**ectified **L**inear **U**nit

RL: **R**einforcement **L**earning

RMSE: **R**oot **M**ean **S**quare **E**rror

RNN: **R**ecurrent **N**eural **N**etwork

SAML: **S**elf-**A**dapting **M**AC **L**ayer

SARSA: **S**tate–**A**ction–**R**eward–**S**tate–**A**ction

SCAFFOLD: **S**tochastic **C**ontrolled **A**veraging for **F**ederated **L**earning

SVM: **S**upport **V**ector **M**achine

TARNet: **T**ask-**A**ware **R**econstruction for **T**ime-Series **T**ransformer

TNC: **T**ime **N**eural **C**onnections

TS-TCC: **T**ime **S**eries - **T**ransformation and **C**ross-**C**orrelation

TST: **T**ime **S**eries **T**ransformer

WSN: **W**ireless **S**ensor **N**etwork

List of Figures

Contents

General Introduction

Background and Motivation

In recent years, wireless sensor networks (WSNs) have become rapidly used in various domains like healthcare, environmental monitoring, and industry. And with the rise of the Internet of Things (IoT) and smart city initiatives, their importance became even more significant. These technologies accelerate the collection of timely data and its processing, which leads to informed decisions and improved operational effectiveness. However, deploying WSNs in any domain comes with various challenges, especially in terms of energy consumption, Since energy has a crucial role in the lifespan and the reliability of these networks. One of the biggest challenges is the energy limitation of the battery-powered sensor nodes, which generally have very limited resources, thus, the need for innovative solutions to optimize their performance under energy constrained environments and extend their operational life.

Problem Statement

Energy Management is a challenging issue in WSNs and the IoT networks. Various Traditional techniques were proposed, like duty cycling, where sensor nodes alternate between the active and sleep modes, and other communication protocols like LEACH, these methods have shown limitations in terms of data reliability and accuracy. The creation of data gaps during sleep periods requires data imputation techniques to keep the data integrity and continuity. Moreover, the final challenge is to balance energy efficiency, network performance and data quality. Current research is exploring how combining duty cycling with control over sensor node mobility can further improve energy efficiency. The development of robust, energy efficient techniques with the help of machine learning models that perform effective data imputation is the solution for achieving the full potential of WSNs in the growing IoT ecosystem.

Objectives of the study

To address these challenges in WSNs, we aim throught this study to:

- Understand the fundamentals of WSNs, and their energy consumption challenges.
- Understand what affects the energy consumption in WSNs and how traditional techniques are used for such tasks.
- Have an overview about AI and Machine Learning, and how it is employed for energy conservation tasks.
- Grasp the principles of Graph Convolutional Networks (GCN), and Federated Learning.
- Review existing work on energy optimization in IoT an WSNs using various machine learning techniques, with a comparison in terms of effectiveness.

Structure of the report

This report is organized as follows:

Chapter 1: We will start by explain the basic concepts and theoretical foundations that WSNs and energy efficiency rely on, that is followed by an overview of how both, traditional and modern techniques can be used to solve energy conservation problems in WSNs.

Chapter 2: In this chapter, we will explain in detail the theoretical and technical architecture of Graph Convolution Networks and Federated learning, their variyants, and how we can apply them.

Chapter 3: Finally, we will review the newest models and techniques that tackle our problematic, the review will be detailed including the key innovations in each work, the overall architecture, and, a critical evaluation of the method.

This thesis therefor aims to provide the reader with a comprehensive understanding of various machine learning algorithms and models used to address energy conservation and prediction problems in wireless sensor networks. We'll approach this from two distinct perspectives: graph theory and time series analysis. Our review will examine how machine learning approaches can further optimize energy efficiency and enhance predictive accuracy in wireless sensor networks, including specific case studies, performance evaluations of various algorithms, and a critical discussion of implications for future research and practical applications in the field.

Chapter 1

Wireless Sensor Networks and Energy Efficiency

This chapter reviews energy efficiency in WSNs, explaining the need for effective energy management due to the power limitations of sensor nodes. It begins by discussing the main factors contributing to energy consumption, such as transmission power and communication overhead. Next, it explores traditional energy-saving approaches, including duty cycling, data aggregation, and energy harvesting.

The chapter also introduces modern techniques, particularly the use of ML, to optimize energy usage and extend network lifetime. The goal is to provide a clear understanding of how these methods address the challenge of prolonging the operational life of WSNs.

1.1 Basic Concepts

1.1.1 Overview of Wireless Sensor Networks

Wireless Sensor Networks (WSNs) are distributed systems of sensor nodes, each node is a small and autonomous device that has a microprocessor, a wireless communication module and it also comes with various sensors(temperature, motion, cameras..). These nodes perform the task of collectively gathering information from their environment, then they transmit this data to a central gateway through a multi-hop communication, the data is then forwarded to a central processing unit for further analysis and decision making. Figure 1.1 illustrates the structure of a usual WSN

We can model the Energy consumption E of each sensor node as:

$$E = E_{\text{Tx}} + E_{\text{Rx}} + E_{\text{comp}}, \tag{1.1.1}$$

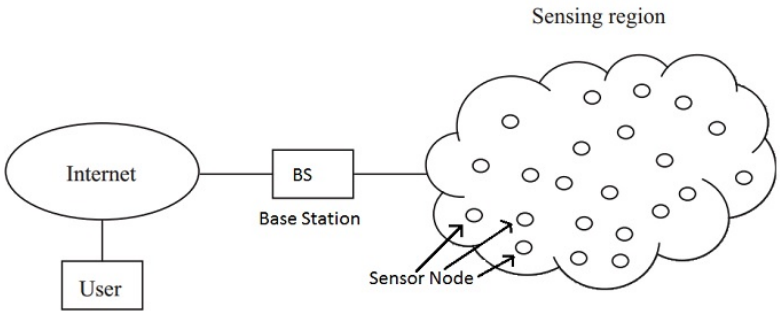

Figure 1.1: General architecture of a Wireless Sensor Network

where E_{Tx} is the energy consumed during transmission, E_{comp} during computation, and E_{Rx} during reception .

Transmission power, distance between nodes, and data processing requirements are the main factors that affect the energy consumption of each node.

1.1.2 Importance of Energy Efficiency in WSNs

Due to the limited battery life of sensor nodes, energy conservation became a crucial concern in WSNs, especially that sensor nodes are usually found in remote and inaccessible areas that makes the replacement of batteries impractical and time consuming, most of the battery life goes to communication processes since the energy required to transmit a bit of data is much higher than one used for computation. The importance of energy efficiency is evident in the fact that the operational life of the network can be extended by optimizing energy usage during the sensing, data processing and communication phases.

Figure 1.2 illustrates the energy consumption breakdown in a typical WSN, showing the need for energy-efficient communication protocols.

1.1.3 Key Challenges in Energy Management

Several challenges are encountered when managing energy consumption in WSNs, including computational costs, communication overhead and the trade off between the network performance and energy saving, the high frequency of data transmission and reception make the communication overhead the primary source of energy drain in WSNs. To reduce the amount of data transmitted, several techniques emerged such as data aggregation, compression and clustering.

Another major challenge is the trade-off between the network performance and the energy conservation, if we reduce the transmission power we can save more energy but on the other hand we may result in a decreased communication range and a higher latency

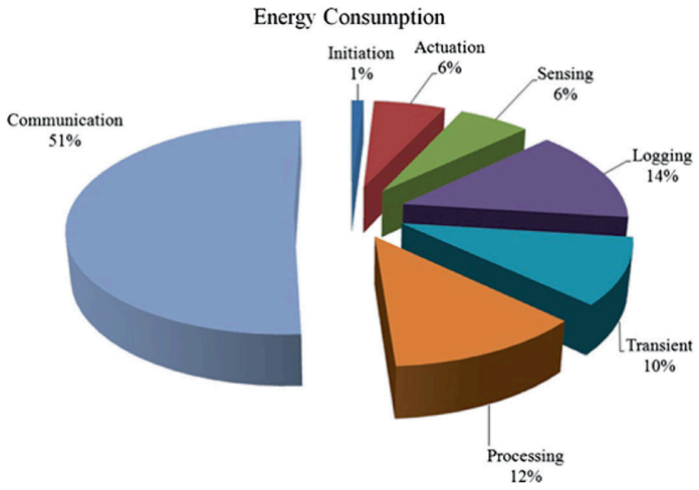

Figure 1.2: Energy Consumption Breakdown in a typical WSN

costing us the network reliability and data delivery rates.

Computational costs are another critical challenge, including the energy used in algorithm execution and data processing, to effectively manage those costs and to prevent premature depletion of node batteries, which could lead to network partitioning and reduced data reliability it is essential to use advanced routing protocols that balance energy consumption across the nodes network.

1.2 Traditional Energy-Saving Techniques:

In Figure 1.3 we can see the breakdown of the different methods that can be used to save energy in WSNs, each focusing on a different perspective, we will discuss some of the main strategies further in this section.

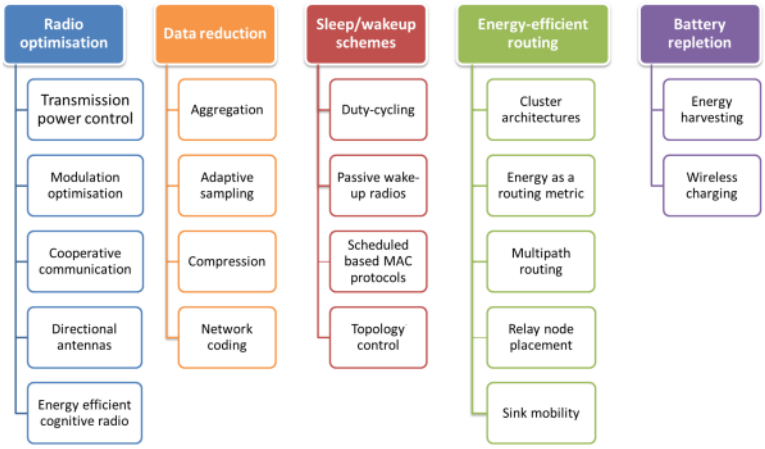

Figure 1.3: Energy saving techniques in WSNs

1.2.1 Duty Cycling

One of the fundamental energy-saving mechanisms in WSNs is Duty cycling, where sensor nodes are periodically switching between the active and sleep states, this method enables the prolongation of the useful life of a network of small battery-powered devices (sensor nodes), allowing those energy constrained devices to monitor physical or environmental conditions for an extended period of time.

Duty cycling addresses this challenge by reducing the amount of time nodes spend in high-powered activity modes. Essentially, nodes alternate between short periods of activity and longer periods of low-power sleep. This approach reduces power consumption significantly, as nodes consume much less power when they are in sleep mode.

We can define the duty cycle as follows:

$$\text{Duty Cycle} = \frac{T_{\text{active}}}{T_{\text{active}} + T_{\text{sleep}}} \tag{1.2.1}$$

where T_{active} is the active time and T_{sleep} is the sleep time.

Duty cycling balances between energy conservation and network performance. When the duty cycle decreases the energy consumption is also reduced, but that costs the network a potential data loss and an increased latency. We can express this relationship as follows:

$$E_{\text{total}} \propto \text{Duty Cycle} \tag{1.2.2}$$

$$\text{Latency} \propto \frac{1}{\text{Duty Cycle}} \tag{1.2.3}$$

where E_{total} represents the total energy consumption. For example, if we consider a sensor node that consumes 50 mW when in active state and 1 mW when sleeping. With a 10 percent duty cycle, its average power consumption is going to be:

$$P_{\text{avg}} = (50 \text{ mW} \times 0.1) + (1 \text{ mW} \times 0.9) = 5.9 \text{ mW} \tag{1.2.4}$$

This represents a great reduction compared to the 50 mW that would be consumed in constant active state. Researchers have proposed a variety duty cycling protocols to effectively implement this technique:

1. Synchronous protocols (e.g: S-MAC, T-MAC): in this case, the sleep and awake periods are synchronized between neighboring nodes ensuring communication by making them awake simultaneously. For example the time is divided into fixed length frames

in Sensor-MAC, starting with a synchronization period then a data transmission period.

2. Asynchronous protocols (e.g: B-MAC, X-MAC): Here nodes are allowed to have independent sleep schedules, communication is usually ensured using techniques like preamble sampling.

3. Hybrid protocols (e.g:Z-MAC): These protocols combine features of both synchronous and asynchronous approaches to adapt to varying network conditions. Z-MAC, for instance, behaves like CSMA under low contention and TDMA under high contention.

1.2.2 Data Aggregation and Compression

Data aggregation and compression are two fundamental techniques to save energy in WSNs by reducing the amount of data exchanged over the network. These techniques exploit spatial and temporal correlation of sensory information in reducing redundancy and optimizing communication .

Aggregation is the process of collecting data from multiple sensors or multiple readings by one sensor to minimize the overall amount of data transmitted. We can achieve this with very good results in WSNs because of the natural correlation between the data gathered by sensors in proximity of each other or readings at consecutive times. Data aggregation can be represented as:

$$D_{\text{agg}} = F(D_1, D_2, ..., D_n) \tag{1.2.5}$$

where D_{agg} is the aggregated data, F is the aggregation function, and $D_1, D_2, ..., D_n$ are the individual sensor readings.

On the other hand, data compression can be divided into 2 methods:

1. Lossless compression: Here the original data is perfectly reconstructed with techniques like Huffman coding where shorter codes are assigned to frequent symbols or like Lempel-Ziv-Welch which Builds a dictionary of repeated patterns.

2. Lossy compression: Here we can achieve a higher compression rate but with some loss in information, techniques that follow this method include Transform coding which Applies transforms like DCT (Discrete Cosine Transform) to represent data more compactly or Predictive coding that uses predictions based on previous values to encode the differences

1.2.3 Energy Harvesting:

Energy harvesting is a technique that allows sensor nodes in WSNs to collect energy from surrounding sources int the WSNs' environment, this provides renewable energy sources to recharge the sensors' battery allowing an extended lifetime of the WSN.

There are various sources used to harvest energy from in WSNs, like solar energy, a very common technique for outdoor WSNs, Thermal energy, Vibration energy and even Radio Frequency energy.

The energy balance of a harvesting-enabled sensor node can be seen as:

$$E_{\text{available}}(t) = E_{\text{initial}} + \int_0^t (P_{\text{harvested}}(\tau) - P_{\text{consumed}}(\tau))d\tau \tag{1.2.6}$$

where $E_{\text{available}}(t)$ is the available energy at time t, E_{initial} is the initial energy, $P_{\text{harvested}}(\tau)$ is the harvested power, and $P_{\text{consumed}}(\tau)$ is the power consumed by the node.

On the other hand, Energy-neutral operation is a key goal in energy harvesting WSNs, and it can achieved when:

$$\lim_{t\to\infty} \frac{1}{t} \int_0^t P_{\text{harvested}}(\tau)d\tau \geq \lim_{t\to\infty} \frac{1}{t} \int_0^t P_{\text{consumed}}(\tau)d\tau \tag{1.2.7}$$

This condition ensures that, on average, the harvested energy is sufficient to power the node's operations indefinitely.

By combining various techniques like duty cycling and energy harvesting we can extend the lifetime of WSNs and save energy, but that is not simple and it requires many design and hardware adjustments, this presents a challenge for other fields like mechanical and electrical engineering.

1.3 Modern Approaches for Energy Saving in WSNs:

1.3.1 Introduction to Machine Learning in WSNs:

Machine learning is a powerful technique that has a large variety of algorithms that can perform different complex tasks in an optimal way after learning from the given data, and within the IoT ecosystem and especially in WSNs, we can find different ways to save energy by addressing different process and optimizing, and for that we can choose suitable ML approaches to analyze complex patterns in the network traffic, sensory data, and the overall energy consumption to make smart decisions that optimize the energy effeciency.

ML is applied for different tasks in WSNs including:

- Predictive maintenance, where the ML model predicts where maintenance is needed before the failure of sensors.
- Adaptive duty cycling, here the model can predict the optimal way of switching sensors on and off to conserve energy.
- Smart Routing, ML can choose the best paths for data transmission, this minimises the communication process.
- Anomaly detection: By identifying unusual behavior (like a malfunctioning sensor), the model can act fast to prevent bigger energy losses.

These applications are deployed using different ML models, including supervised, unsupervised and semi-supervised learning, in the following section we will give more details about how every model works and what problematic is addresses.

1.3.2 Supervised Learning Approaches:

In supervised learning the model is trained on a labeled dataset (inputs: features, and known outputs: target) , the model learns the relationships between the inputs and the outputs, to use it later to predict unknown output. Supervised learning is used a lot to solve problems in the context of WSNs, like, localization and objects targeting, event detection and query processing, media access control , security and intrusion detection... Common ML algorithms include:

Decision Trees and Random Forests

Decision trees are used to handle non-linear datasets, and it is used for classification problems. The input data is iterated through a learning tree to predict the label. Random forests are an extension of decision trees where multiple decision trees are aggregated to improve the robustness and generalization. **Application:** Optimizing cluster head selection in hierarchical WSNs by considering factors like residual energy, node density, and distance to the base station.

Figure 1.4 shows the decision tree classifier used to select the optimal MAC algorithm in an SAML architecture proposed by Sha et al. in “Self-Adapting MAC Layer” .

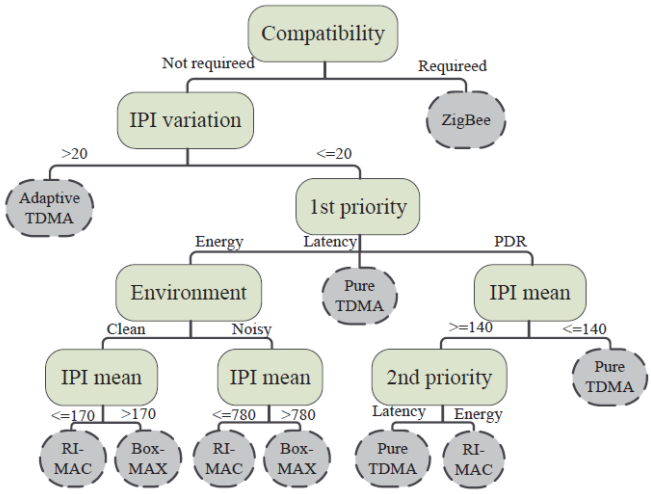

Figure 1.4: A Decision Tree classifier

Support Vector Machines (SVMs)

SVMs is a classification model that learns to find the optimal hyperplane or kernel that separates the classes in the feature space, maximizing the gap between different classes, and new input will be classified based on which side of the gaps are on

Application: Classifying network traffic patterns to predict congestion and adjust transmission power, other application include security and localization.

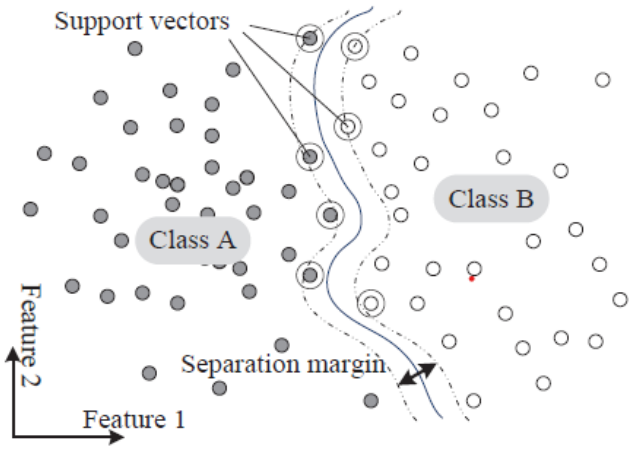

Figure 1.5: Non-linear support vector machines

Figure 1.5 is an illustration of a WSN's data as points in the feature space, with a non-linear SVM as a divider.

Neural Networks and Deep learning:

Neural networks are built by layering multiple chains of perceptrons(neurons), those neurons are decision units that by combining them the algorithm can learn very complex and non-linear functions, deep learning uses NN in-depth for more complex pattern recognition, however this technique is very expensive computationally which limits their usage for WSNs, but with the right model we can optimize the energy consumption in WSNs.

Application: Node localization, Predicting energy consumption of sensor nodes based on various environmental and network-related features. Deep learning models were also used to optimize routing protocols, data compression, and adaptive sampling techniques in WSNs.

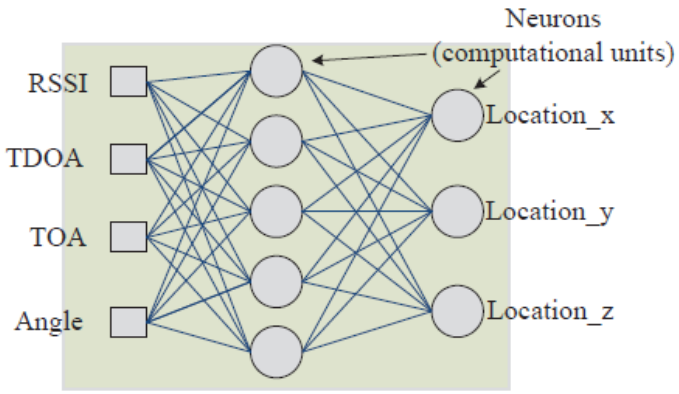

Figure 1.6: A Simple Neural Network For Node Localization

Figure 1.6 shows an example of node localization in WSNs in 3D space using supervised neural networks.

Recent trends in deep learning for WSNs include:

- Convolutional Neural Networks (CNNs) used for spatial feature extraction in sensory data.
- Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks mostly applied for temporal pattern recognition in energy consumption
- Attention mechanisms used to focus on the most relevant features in energy optimization tasks.

1.3.3 Unsupervised Learning Approaches

In unsupervised learning, the model is not given labels. In essence, the goal of supervised learning is to classify the given sample data into groups from the patterns identified in that

given sample, this techniques offers an opportunity for energy saving in WSNs without the need for labeled data, the most common approaches are:

K-means Clustering

The algorithm divides the given data points into K clusters, it is widely used in sensor nodes clustering for its simple implementation and linear complexity.

Application: As already mentioned, K-means clustering can be used in grouping sensor nodes with similar energy consumption patterns to implement targeted energy-saving strategies. K-means has also been used for energy-efficient data aggregation and compression in WSNs.

Recent advancements in clustering for WSNs include:

- Adaptive K-means algorithms which automatically determine the optimal number of clusters
- Fuzzy C-means clustering to handle overlapping energy consumption patterns
- Spectral clustering in energy-efficient topology control in WSNs

Principal Component Analysis (PCA)

This is a multivariate algorithm for dimentionality reduction and also data compression by extracting valuable information from the data and then transforming it to a new set of variables known as principal components.

Application: Compressing sensor readings to reduce transmission energy without a great loss in information. PCA has also been used for anomaly detection in energy consumption patterns.

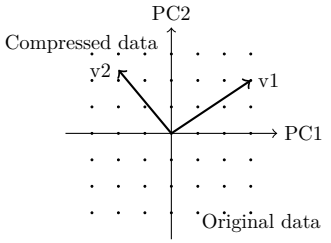

Figure 1.7: PCA for Data Compression in WSNs

Recently PCA was used in WSNs as:

- Kernel PCA for handling non-linear relationships in sensor data

- Incremental PCA for online data compression in streaming sensor data
- Robust PCA for handling outliers and noise in energy consumption data

1.3.4 Semi-supervised learning

Semi-supervised learning uses a combination of labeled and unlabeled data to train the model, this is so useful in WSNs since fully labeled data is hard to obtain in IoT environments, common semi-supervised techniques include:

Label Propagation

Label propagation works o graph structured data, where node can be labeled and unlabeled, and edges are the relationships between them. The model propagates labels from from labeled nodes to the unlabeled ones based on the similarity between every 2 nodes, we can find this similarity as edge weights, this process is done repetitively until we reach a global consensus.

Application: Classifying energy states of sensor nodes with limited ground truth data. Label propagation has also been used for energy-efficient event detection and localization in limited labeled WSNs.

Co-training

Co-training is a technique where multiple classifiers are trained on different 'views' of the data assuming that each view is independent, each classifier is trained on some of the features, later their predictions of the unlabeled data is used to train other classifiers that complement eachother.

Application: Combining network topology and sensor reading features to predict optimal transmission power levels. Co-training has also been applied to energy-efficient data aggregation and fusion in heterogeneous WSNs.

Graph Neural Networks (GNNs)

GNNs are neural networks created to work specifically on graph-structured data, which makes their implementation in WSNs natural where sensor nodes and their links are easily represented as graphs. GNNs takes into consideration both node features and the graph topology to learn node embeddings.

The graph is represented by and adjacency matrix A that contains the edges and the edges' weights , and the feature matrix X that has the features of every node i.

The hidden representation of a node i at $t+1$ can be obtained from the message passing process which is the sum of the edge weights between the nodes i and every neighbor j multiplied by the transformation function of the representation of the node i at t , all multiplied by a non-linear activation function (like ReLU), this process repeats for T iterations, so that information can propagate across the graph. After T iterations, the node embeddings $H^{(T)}$ can be used for various tasks such as node classification or link prediction.

Application: In the context of WSNs, GNNs can be used for tasks such as energy-efficient routing, where the graph structure of the network is designed to make intelligent routing decisions. By learning to aggregate information from neighboring nodes, we can predict energy consumption patterns, detect anomalies, and optimize communication in a distributed way.

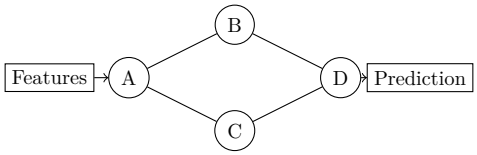

Figure 1.8: A Simple GNN for WSN Optimization

Recent advancements in GNNs, such as attention mechanisms and spatio-temporal GNNs, further optimize their usage in WSNs. Attention-based GNNs assign different weights to different neighbors during the aggregation process, this makes the model focus on the most relevant nodes in the network. Spatio-temporal GNNs, on the other hand, model both spatial and temporal dependencies in the sensory data, which makes the suitable for dynamic WSNs where energy consumption patterns change over time.

1.3.5 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning technique that uses the trial and error process with rewards and penalties to achieve optimal results. Recently, in WSN energy optimization, Rl was widely used due to its capacity to learn the most optimal policies by interacting with the WSN environment, one of the most used RL techniques is Q-learning, where the agent constantly updates the total rewards (Q-value) received from taking an an action a_{t} at a certain state s_{t}, this method can be visualized in Figure 1.9 : **Application:** Adaptive duty cycling, where RL agents learn to adjust the sleep/wake schedules of sensor nodes based on network conditions and energy availability. RL has also been used to optimize routing decisions and transmission power control in WSNs.

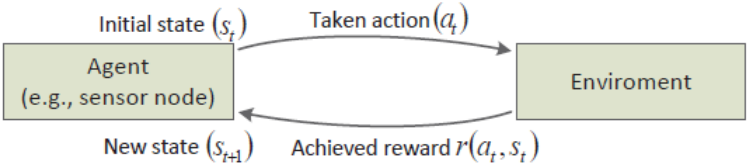

Figure 1.9: Q-learning method

Recent research in RL for WSNs brought advancements including:

- Deep Reinforcement Learning (DRL) combining deep neural networks with RL for handling high-dimensional state spaces
- Multi-agent Reinforcement Learning (MARL) for coordinated decision-making among multiple sensor nodes
- Model-based RL approaches that learn a model of the environment to improve sample efficiency

1.3.6 Challenges of ML in WSNs:

While the necessity of deploying ML models in WSNs is evident for its significant improvement in minimizing the energy consumption, it is necessary to acknowledge the challenges and difficulties so we can address them in future works. One of the main problems is the resource constrained environments, due to the resource limited nature of sensor nodes, it is hard to implement complex ML models in such environments, in the other hand, the ML models face a challenge in adapting to dynamic network conditions and energy patters, future works should also address the scalability problem to ensure that the ML solutions can adapt to large scale WSN deployments, and while ML models deliver outstanding results, they are often seen as black boxes, so it is important to make those solutions understandable to allow further advancements from other researchers.

Conclusion

In this chapter, we provided all the necessary basic knowledge in the scope of WSNs and energy efficiency and we also reviewed various strategies to improve it, from traditional techniques like duty cycling and data aggregation to advanced ML-based methods. While these approaches are effective in reducing energy consumption, their success often depends on challenges like limited computational resources and network dynamics. Combining traditional approaches with ML-based techniques could offer significant solutions for improving the sustainability and reliability of WSNs in the years ahead.

Chapter 2

Federated Learning and Graph Convolutional Networks

In this chapter, we will explain two important concepts that are changing data processing and machine learning (ML) in wireless sensor networks (WSNs) and IoT will look like: Federated Learning (FL) and Graph Convolutional Networks (GCNs).

Federated Learning is a new approach that appeared first in 2017, this technique allows us to train machine learning models without the need to a centralized data center. This is very pertinent for IoT environments and WSNs, where we want to save energy and protect privacy because they contain sensitive informations. We will explain how FL works in details, the architecture, and all the components and the algorithms involved in the process.

On the other hand, GCNs are a type of neural network that can work with data that can be represented with a graph topology (nodes and edges). It is evident that we can use this model type for wireless sensor networks, where each sensor is connected to others. We will explain how these networks work and how they can help us process data from our sensors in a smart way.

By the end of this chapter, you will have all the necessary knowledge to understand any work and any architecture that involves the usage of FL and GCNs.

2.1 Graph Convolutional Networks (GCNs):

2.1.1 Definition:

Graph convolutional networks are a class of neural networks that were specially designed to work on graph-structured data.These networks develop the idea of convolutional neural

networks from regular grid-like structures to irregular graph domains. In Wireless Sensor Networks, GCNs provide a promising approach for processing and analyzing this kind of data, which is highly complex and has spatial connections. A GCN can be formally defined as a function$f(X, A)$, with $X \in \mathbb{R}^{N \times D}$a feature matrix of N nodes having D-dimensional feature each, and $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix describing the structure of the graph. GCN learns a transformation of these inputs to produce node or graph level representations:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2.1.1}$$

where $\tilde{A} = A + I_N$ includes self-connections, $\tilde{D}$ is its degree matrix, $W^{(l)}$ is a weight matrix, and σ is a non-linear activation.

2.1.2 Graph Theory fundamentals:

To fully understand the power of GCNs in domains like WSNs, we will start by explaining the basic concepts of graph theory in this section:

Nodes and Edges:

A graph $G = (V, E)$ consists of vertices (nodes) V and edges E. In WSNs, nodes represent sensors, and edges represent connections or relationships, like proximity or communication links.

Edges can be:

- **Undirected**: $(i, j) \equiv (j, i)$:used for symmetric relationships.
- **Directed**: $(i, j) \neq (j, i)$:for asymmetric flows.
- **Weighted**: Every edge has a weight w_{ij} representing the strength of the connection.

Graph Representations:

The common way to represent a graph is with a :

1. **Adjacency Matrix**: A square matrix $A \in \mathbb{R}^{N \times N}$ where $A_{ij} = 1$ if an edge exists, or $A_{ij} = w_{ij}$ in a weighted graph.
2. **Laplacian Matrix**: Defined as $L = D - A$, where D is the degree matrix. It is mostly essential in spectral graph theory.

3. **Incidence Matrix**: $B \in \mathbb{R}^{N \times M}$ for a graph with M edges, defined as:

$$B_{ij} = \begin{cases} 1 & \text{if node } i \text{ is the source of edge } j \\ -1 & \text{if node } i \text{ is the target} \\ 0 & \text{otherwise} \end{cases} \tag{2.1.2}$$

These representations help GCNs to use the network topology in the learning process.

2.1.3 GCN Architecture:

The main idea behind GCN is to apply convolution on a graph instead of a 2-d array (usually representing pixels) , similarly to Neural networks, GCN has an input layer, an output layer and a variant number of hidden layers, the key difference is in the input, where for a GCN is a graph (a set of x) and for a NN is a single input x.

Graph Convolution:

Graph Convolution in the most basic idea behind all the GNN architectures, it is a function that predict the features of the node in the next layer based on its neighbours' features, Two principle approaches for graph convolution are:

1. **Spectral Approach**:The convolution operator is defined in the spectral domain using the Fourier transform, first the graph signal is transformed to the spectral domain, then we conduct an element wise multiplication resulting in the convolution operation, the resulting signal is the transformed back using the inverse Fourier, we can define this process using the graph Laplacian:

$$g_\theta \star x = U g_\theta(\Lambda) U^T x \tag{2.1.3}$$

   where U is the matrix of eigenvectors of the normalized graph Laplacian, Λ is the diagonal matrix of its eigenvalues, and $g_\theta(\Lambda)$ is a learnable filter in the spectral domain.

2. **Spatial Approach**: The convolution is defined directly on the graph topology, it aggregates information from a node's local neighborhood with a permutation-invariant function and updates, a general form of spatial graph convolution can be expressed as:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} W^{(l)} h_j^{(l)}\right) \tag{2.1.4}$$

where $h_i^{(l)}$ is the feature vector of node i at layer l, $\mathcal{N}(i)$ is the neighborhood of node i, α_{ij} are attention coefficients, and $W^{(l)}$ is a learnable weight matrix.

Layer-wise Propagation:

The mathematical rule used to compute a hidden layer output from the last layer output is called a propagation rule, there are different variations of f but we can generalize the propagation rule for GCNs as:

$$H^{(l+1)} = f(H^{(l)}, A) \tag{2.1.5}$$

This enables nodes to exchange information along the graph edges, aligning well with the distributed nature of WSNs.

2.1.4 Key Components of GCNs

Graph Laplacian

The normalized Laplacian is used in the graph convolutional layer and it is the core of the GCN since it is responsible for performing the convolution operation, it can be defined as:

$$L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{2.1.6}$$

This operator models the passage of information across the graph, helping identify the most impactfull nodes or clusters.

Aggregation Functions:

For every node in the graph, the features are updated after taking the weighted average of its neighbors' features, the weights are usually found in the normalized adjacency matrix,we can use different aggregation functions like:

- Mean aggregation
- Max aggregation
- Attention-based aggregation, where attention coefficients prioritize important neighbors.

Non-linear Activation:

The aggregated new features are non-linearly transformed using activation functions like ReLU or sigmoid allowing the network to capture patterns in the graph, the output is the final node representation.

2.1.5 Variants of GCNs

Many variants of GCNs have been developed to address different challenges or improve performance in different cases. These variants can be also useful in optimizing GCNs for WSN applications.

Graph Attention Networks (GATs)

The main idea introduced but GAT is to compute a certain coefficient that is derived from the degree matrix of the graph, in a normal GCN it is computed explicitly and multiplied by the node's feature projection, however, in GATs this coefficient is computed implicitly and is used as an additional information to find the most important nodes, this allows the model to assign different importances to different neighbors when aggregating information. The attention coefficients are computed as:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})} \tag{2.1.7}$$

where $e_{ij} = a(Wh_i, Wh_j)$ is a learned attention function. This approach can be particularly beneficial in WSNs, where some sensor connections may be more informative than others, leading to more efficient use of network resources.

GraphSAGE

GraphSAGE was proposed as a framework in a paper where the main contributions are training the model in an unsupervised manner allowing the model to generate embeddings for unseen nodes. The key idea is to learn a set of aggregator functions that can be applied to any node's neighborhood:

$$h_i^{(l+1)} = \sigma(W^{(l)} \cdot \text{CONCAT}(h_i^{(l)}, \text{AGG}(\{h_j^{(l)}, \forall j \in \mathcal{N}(i)\}))) \tag{2.1.8}$$

This approach is particularly relevant for dynamic WSNs, where new sensors may be added to the network over time.

ChebNet

ChebNet uses Chebyshev polynomials to approximate spectral graph convolutions, this brings a good balance between efficiency and expressiveness:

$$g_\theta \star x \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{L}) x \tag{2.1.9}$$

where T_k are Chebyshev polynomials and $\tilde{L} = 2L/\lambda_{\max} - I_N$ is the scaled Laplacian. This approach can be effective in capturing multi-scale features in WSNs, potentially leading to more accurate and energy-efficient data analysis.

2.1.6 Applications of GCNs

GCNs are applicable in WSNs for:

- **Topology Optimization**: Learning optimal network topologies to balance data quality and energy use.
- **Data Compression**: Reducing transmitted data by learning efficient representations.
- **Distributed Learning**: Enabling local processing across sensors to reduce central processing load.

2.2 Federated Learning:

2.2.1 Definition:

Federated learning, which can also be called collaborative learning, is a sub field of machine learning, it was specifically designed to solve the problems related to data privacy and communication efficiency in decentralized systems. It was first introduced by McMahan et al. (2017), FL aims to train machine learning models on various decentralized datasets contained in local nodes, that prevents the direct access to the raw data and the data exchange between nodes.

Basically, FL works with the same principle of collaborative learning, where we have a global model that is trained across several devices (or nodes), we call them: the clients. The data remains locally in every client and the computations are performed on the node locally with the model updates shared with the central server . We can define the FL approach as an optimization problem:

$$w^* = \arg\min_{w} \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \tag{2.2.1}$$

Where, w^* represents the optimal model parameters, K is the total number of clients, n_k is the number of samples on the k-th client, n is the total number of samples among all clients, and $F_k(w)$ is the local objective function for the k-th client. This equation summarizes the core of FL: a global objective attained by the weighted aggregation of local objectives.

2.2.2 Federated Learning Architecture:

FL has mainly two main components: a central server and a network of edge devices also known as clients.

Central Server and Edge Devices

The main role of the central server is to perform the aggregation, but it is also responsible for other tasks including:

Model Initialization: the initial model architecture and parameters are selected based on the task at hand.

Client Selection: For each round of training, the server can choose a subset of clients based on characteristics such as data quality or computational capacity, iteratively, or randomly.

Global Model Maintenance: The server aggregates model updates received from clients and maintains the global model. This includes implementing aggregation algorithms such as FedAvg (McMahan et al., 2017) or more advanced techniques like FedProx (Li et al., 2020).

Convergence Monitoring: The global model's performance is tracked to then determine when to terminate the training process based on predefined convergence criteria or a maximum number of rounds.

And for the edge devices, which in a WSN perspective can be a single sensor or a cluster of sensors, they are responsible for Collecting, storing, and preprocessing local data. And even, handling data augmentation, normalization, and feature extraction as necessary, training the global model on their local datasets. After local training, devices compute model updates. These updates can be in the form of gradients, model parameters, or other representations to minimize communication overhead.

The communication between these two components follows these steps:

Let w^t be the global model at round t, and w_k^t be the local model of client k at round t. The server-client interaction in round t can be described as:

1. Server broadcasts w^t to selected clients.

2. Each selected client k updates its local model

3. Clients send updates to the server.

4. Server aggregates updates: $w^{t+1} = w^t + \eta \sum_{k=1}^{K} \frac{n_k}{n} \Delta w_k^{t+1}$

where η is the server learning rate, n_k is the number of samples on client k, and $n = \sum_{k=1}^{K} n_k$ is the total number of samples.

In short, the Server starts by choosing clients and sending the base model, then the clients train the model on their local data and send the model updates to the server where the updates are aggregated and then sent again to the clients, this is done until a certain criteria is met or until a certain number of training rounds is reached.

Local Training and Aggregation

We will now dive deeper in the local training process,

1.First, when receiving the global model, every device can adapt it to its local data distribution using techniques like fine-tuning or meta-learning approaches.

2. Then, the devices compute the loss on their local dataset using an appropriate loss function. For a classification task, for example the cross-entropy loss.

3. Later, devices compute the gradients of the loss respecting the model parameters. For neural networks, this usually is done with backpropagation.

4. Then, the local model parameters get updated using an optimization algorithm such as Stochastic Gradient Descent (SGD).

5. After several epochs of local training, devices compute the model update, usually it is the difference between the final local model and the first global model

The aggregation process, done by the central server, combines updates from several clients to improve the global model. The most common aggregation method is Federated Averaging (FedAvg), which can be expressed as:

$$w^{t+1} = w^t + \eta \sum_{k=1}^{K} \frac{n_k}{n} \Delta w_k^{t+1} \tag{2.2.2}$$

where η is the server learning rate, which can be adjusted to control the impact of client updates on the global model.

More advanced aggregation techniques have been proposed to address challenges such as non-IID data and system heterogeneity:

1. **FedProx (Li et al., 2020):** Adds a proximal term to the local objective function to limit the impact of client drift:

$$\mathcal{L}_k^{\text{FedProx}}(w) = \mathcal{L}_k(w) + \frac{\mu}{2}\|w - w^t\|^2 \tag{2.2.3}$$

2. **SCAFFOLD (Karimireddy et al., 2020):** Introduces control variants to correct for client drift:

$$w^{t+1} = w^t - \eta(\frac{1}{K}\sum_{k=1}^{K}\Delta w_k^{t+1} - c_k + c) \tag{2.2.4}$$

where c_k and c are client and server control variants, respectively.

3. **FedDyn (Acar et al., 2021):** Dynamically adjusts the impact of each client's update based on its consistency with the global objective:

$$w^{t+1} = w^t - \eta(\nabla f(w^t) + \frac{1}{K}\sum_{k=1}^{K}(\nabla f_k(w_k^{t+1}) - \nabla f_k(w^t))) \tag{2.2.5}$$

These advanced aggregation methods aim to improve convergence speed and model performance, particularly in heterogeneous environments common in WSNs.

Communication Efficiency

Communication in FL is based on sequential exchanges of model updates between edge devices and the central server which consumes a lot of energy in WSNs. The energy consumption in one communication round can be expressed as: $E_{\text{comm}} = \sum_{k=1}^{K}(E_{\text{tx}}(|\Delta w_k|) + E_{\text{rx}}(|w|))$ where E_{tx} and E_{rx} are the energy consumed in transmitting and receiving, and, $|\Delta w_k|$ is the size of the model update from client k, and $|w|$ is the size of the global model.

To enhance the communication efficiency in a FL environment we can consider:

- **Gradient Compression**: It reduces the size of the gradient updates sent by the clients to the server which effectively lowers communication overhead. Techniques like quantization, sparsification, and gradient sketching are the most used ones, allowing devices to communicate smaller updates without a major loss in model accuracy.
- **Model Pruning**: Here, we remove the less important parameters from the model, which reduces the size of the updates transmitted. By pruning small weights, the

communication payload is lowered, this conserves energy in resource-constrained environments like WSNs.

- **Knowledge Distillation**: In this technique, a smaller model is trained to mimic the behavior of a larger, more complex model. By communicating this smaller model during the learning process.
- **Federated Dropout**: This was mainly inspired by dropout techniques in deep learning, it only transmits a subset of model parameters during each communication round.

All these techniques aim to lower the energy consumed in the communication while also keeping a high performance.

Model updates

Model updates are at the essence of the FL process, and their optimization is important to both, improving learning performance and conserving energy in WSNs. The way and frequency in which the updates are exchanged between clients and the server impacts on the system's efficiency directly. Most used techniques to optimize the model updates are:

- **Periodic Averaging**: Instead of communicating updates after every local epoch, clients can perform multiple local training iterations before sending their updates to the server.
- **Importance-based Updates**: In this method, clients only send updates if they exceed a certain threshold of importance. This ensures that only the most significant updates, which are likely to meaningfully impact the global model, are transmitted.
- **Error-Feedback Mechanisms**: Clients can accumulate small, incremental updates and only communicate once the cumulative update exceeds a specified threshold.
- **Compressed Updates**: Here, the model differences can be compressed before transmission, usually using techniques like low-rank approximations or other ways for compression.
- **Adaptive Optimization**: Clients can adapt their local optimization procedures, adjusting learning rates or optimization algorithms based on the global model's performance.

2.2.3 Privacy and Security in Federated Learning

Ensuring data privacy is a the most important role of FL, especially in WSNs where sensitive sensor data is collected.

Differential Privacy

Differential Privacy (DP) is a framework for protecting the privacy of individual data points during the FL process. It adds a degree of randomness into the model updates,with it, the DP can ensure that the presence or absence of any single data point cannot be inferred from the aggregate model.

There are several variants of DP that have been adapted for FL: Client-level DP, Record-level DP, Local Differential Privacy (LDP),Moments Accountant, and many other variants dependinf on the task at hand.

Secure Aggregation

Secure aggregation allows the central server to compute the sum of client updates without accessing the individual updates. This is usually done with cryptographic techniques, like homomorphic encryption or secret sharing.

This adds a layer of protection against possible data leaks or breaches. This is especially important in WSNs where the data being transmitted could be sensitive and require a high level of security.

2.2.4 Applications on FL:

FL was recently popularized in many domains, usually where the data privacy is important, for example, in healthcare facilities where communication and collaboration between various hospitals and clinics is important, we find that now they use FL for predictive modeling for patient outcomes.

Also, in IoT networks like smart homes and so, we can find FL used for anomaly detection because it ensures learning from distributed sources without centralizing sensitive information.

We can also find FL used in the finance sector for tasks like fraud detection, this allows multiple banks to collaborate in training a model without having to share their sensitive client data.

2.3 Integration of FL and GCNs:

In graph-based data, FL can be adapted to handle the interconnections of nodes. The local objective function for each node i in a GCN-based Federated Learning setup can be

formulated as:

$$F_i(w) = \mathcal{L}(f_w(A_i, X_i), Y_i) \tag{2.3.1}$$

where A_i is the local adjacency matrix, X_i are the node features, and f_w is the GCN function parameterized by w. The global objective then becomes:

$$\min_w \frac{1}{N} \sum_{i=1}^{N} F_i(w) \tag{2.3.2}$$

where N is the total number of nodes in the network.

The integration of Federated Learning enhances GCNs in several ways:

1. Privacy Preservation: FL allows GCNs to learn from distributed graph data without centralizing sensitive information.
2. Scalability: By distributing the computation load, larger graph structures can be processed better.
3. Adaptability: The model can adapt to local graph structures while keeping global knowledge.

The enhanced GCN model can be represented as:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2.3.3}$$

where $W^{(l)}$ are now learned through federated optimization.

To implement Federated GCNs in WSNs. A typical architecture consists of:

- Local Nodes: Each sensor node runs a local GCN model.
- Aggregator Nodes: Intermediate nodes that aggregate updates from local nodes.
- Central Server: Coordinates the federated learning process and maintains the global model.

Conclusion:

In conclusion, we have explained in this chapter two key technologies: Graph Convolutional Networks (GCNs) and Federated Learning (FL). We have also covered the basic concepts of GCNs, their various architectures, and the applications they can be used for. We then explored how can FL change the way we do collaborative model training while keeping data private and secure.

The combination of these two technologies—GCNs and FL—opens up a lot of possibilities for future research. By bringing them together, we might find new ways to approach distributed learning, especially in areas like WSNs and IoT applications.

As we move forward, the technologies discussed here are more than just technical innovations. They represent a shift towards more adaptable, efficient, and privacy-focused machine learning systems. This chapter sets the groundwork for new methods and applications that could change how we handle data analysis and model training.

Looking ahead, the challenge will be making the most of these tools, solving the current problems, and figuring out what else they can do when combined. The ideas discussed here provide a starting point for more research and development taking into consideration the huge potential that lies in graph-based learning and decentralized AI.

Chapter 3

Literature Review

This chapter aims to provide a comprehensive overview of the most recent approaches used for energy conservation in WSNs and IoT application with emphasis on duty cycling optimization. This analysis will serve as a foundation for understanding ML and DL architectures that were proposed by various researches, in which we will review each method and propose a critical review of it in this thesis.

3.1 Energy Efficiency in Reinforcement Learning for Wireless Sensor Networks:

In this paper, M.Kozlowski and R.Mcconville proposed an innovative approach tackling energy-effecient indoor localization using RL techniques in WSNs. The method relies in its core on its formulation as a MDP and the application of SARSA algorithm for continuous weak training, in the following sections we will explain the innovation brought by this paper and the key models and techniques used in this work.

3.1.1 Markov Decision Process (MDP) Formulation:

The problem is formulated as an MDP with two states: S1: Enhanced sensing state, and S2: Low-power sensing state. With two actions possible in each state:

A1: Stay in the current state, and A2: Switch to the other state

They also used a simple reward function that is designed to encourage energy efficiency while also maintaining performance:

$$r(s_t, a_t) = \begin{cases} -1 & \text{if } s_t = S1 \text{ and } a_t = A1 \\ +1 & \text{if } s_t = S2 \text{ and } a_t = A2 \\ 0 & \text{otherwise} \end{cases} \tag{3.1.1}$$

This reward function penalizes staying in the energy-intensive state (S1) and rewards staying in the low-power state (S2). And to ensure the localization performance they added an additional performance boost Bt :

$$B_t = \begin{cases} -1 & \text{if } e(t) \geq e(t-1) \\ +1 & \text{if } e(t) < e(t-1) \end{cases} \tag{3.1.2}$$

where e(t) is the localization error at time t. This boost aims to encourage the system to improve or maintain localization accuracy even when in the low-power state.

3.1.2 SARSA Algorithm:

The authors also employed an on-policy temporal difference learning method by using the SARSA algorithm, the objective is to learn the optimal policy, with the Q-value update rule :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[B_t + r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{3.1.3}$$

where: α is the learning rate, γ is the discount factor, $Q(s_t, a_t)$ is the Q-value for state s_t and action a_t, B_t is the performance boost and $r(s_t, a_t)$ represents the immediate reward

The performance boost B_t included in the Q-value update allows the system to balance energy efficiency and localization accuracy dynamically.

3.1.3 Action Selection Methods:

Three action selection methods are used in this paper to find the trade-off between exploitation and exploration:

1. **Greedy**: It always selects the action with the highest Q-value.

$$a_t =_a Q(s_t, a) \tag{3.1.4}$$

2. **-Greedy**: Selects the greedy action with probability 1- , and a random action with

probability .

$$a_t = \begin{cases} {}_aQ(s_t, a) & \text{with probability } 1 - \epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases} \tag{3.1.5}$$

3. Softmax: Selects actions probabilistically based on their Q-values.

$$P(a_t = a) = \frac{e^{Q(s_t,a)/\tau}}{\sum_b e^{Q(s_t,b)/\tau}} \tag{3.1.6}$$

where is the temperature parameter controlling the exploration-exploitation trade-off.

3.1.4 Hidden Markov Model (HMM) for Localization:

For localization inference, M.Kozlowski and R.Mcconville employ a Hidden Markov Model (HMM) approach. The joint probability of locations xt and observations zt is given by:

$$p(x_{1:t}, z_{1:t}) = p(x_0) \prod_{i=1}^{t} p(z_t|x_t) p(x_t|x_{t-1}) \tag{3.1.7}$$

where:

- $p(x_0)$ is the initial state distribution
- $p(z_t|x_t)$ is the emission probability
- $p(x_t|x_{t-1})$ is the transition probability

The emission probabilities are modeled as Gaussian distributions:

$$p(z_t|x_t) = \sum_{k=1}^{K} \mathcal{N}(z_t|\mu_{jk}, \sigma_{jk}) \tag{3.1.8}$$

where j are the location states and k is the access point sensors. This Gaussian model allows a dynamic representation of the signal strength distributions at every location.

Algorithm Implementation:

The authors proposed an algorithm that combines the MDP formulation, HMM based localization and SARSA learning to ensure an energy efficient adaptive localization, the algorithm can be outlined as follows:

Algorithm 1 HMM and SARSA Algorithm

```
Initialize HMM and SARSA parameters
for each time step do
    Collect sensor observations
    Perform localization inference using the HMM
    if in enhanced sensing state (S1) then
        Collect weak labels from oracle sensors
        Re-estimate HMM parameters
        Calculate localization error
    end if
    if in low-power state (S2) then
        Retain previous error estimate
    end if
    Calculate performance boost
    Select next action using chosen selection method (Greedy, ϵ-Greedy, or Softmax)
    Update Q-values using SARSA
    Transition to next state based on selected action
end for
```

3.1.5 Simulation and Performance:

The method was validated on the SPHERE challenge dataset that has 4 unique access points, location labels and sensors that can be used as ”oracles”, the model was trained on 15

The simulation state space size varied between 10 to 30 states with a variable number of access points, and the BLE path loss model was used to simulate signal strength, after deploying the solution in this environment, the experiment showed a good performance on the validation results, with the greed method improved from 4.8m error (Control) to 2.5m error (Reinforced) and it has also reduced the dependence on energy-inefficient sensors to 0, while the -Greedy method showed improvement from 4.8m error (Control) to 2.1m error (Reinforced) and a 16% dependence after 100% of iterations , finally, the Softmax method Improved from 4.8m error (Control) to 1.8m error (Reinforced) with a 35% dependence after 100% of iterations.

This shows that the algorithm successfully reduced dependence on energy-inefficient sensors while improving localization accuracy, with the -Greedy method showing the best results for the action selection, which offers a good trade-off between sensor usage and performance improvement.

3.1.6 Critical review:

While the paper presents an innovative approach by formulating the problem as an MDP with energy aware states and combining RL techniques with traditional localization methods, also the idea of the continuous weak training is impressive an SARSA was well suitable for it.It is important to note the limitations of this method, to be able to further enhance it.

First, the solution is not well suitable for real life application in energy efficient indoor localization, we can deduce this from the simulation and validation environment of the RL agent, where we have the dependence on the oracle sensors, which are high-accuracy but energy-intensive sensors, which is not always the case, also the method does not account for dynamic changes in the environment (e.g furniture moves, new obstacles..) which may affect the RSS patterns, the paper also states that the real world validation only achieves room-level accuracy which is not enough for many applications, Furthermore, the simulation only takes up to 30 states which is relatively small for actual environments.

Aside from the real world application, the method also shows some limitations in terms of methodology, the method was primarily evaluated on simulated data, with only limited validation on a single real-world dataset (SPHERE Challenge) , the performance appears to be sensitive to many parameters (learning rate, discount factor, oracle weights, ...) which were chosen empirically .Also, the method was not compared against other state of art adaptive or energy aware localization techniques, which makes it difficult to evaluate its performance compared to other methods performing the same task, the method could also explore using loss models other than a basic BLE path loss, to fully capture the details and complexities of real indoor environments.

Despite these limitations, this work provides a solid theoretical foundation for future research in energy-aware adaptive localization systems.

3.2 GSAVES:

In this paper, Laidi et al. presents GSAVES (Graph Sensor AdVersarial for Energy Saving), a new approach for maintaining sensor energy in event-based Internet of Things (IoT) applications. The proposed method aims to preserve energy in both sensing and communications by deactivating a portion of IoT devices while using readings from active sensors and the network's spatial correlation to generate missing data. In this solution no duty cycling scheduling is required since the sensor deactivation process is random, meanwhile an adversarially trained graph Convolutional network (GCN) generates the missing data of the sleeping sensors.

The core idea is to extend the network's lifetime without impacting the network's reliability, and that is by accurately estimating the readings of the sleeping sensor after learning the spatio-temporal characteristics of the sensor network . The GSAVES model operates at the data collector level (cloud, base station, or edge), where the generator is proposed to generate the missing data of sleeping sensors. Figure 3.1 illustrates the Overview of the proposed solution in this paper.

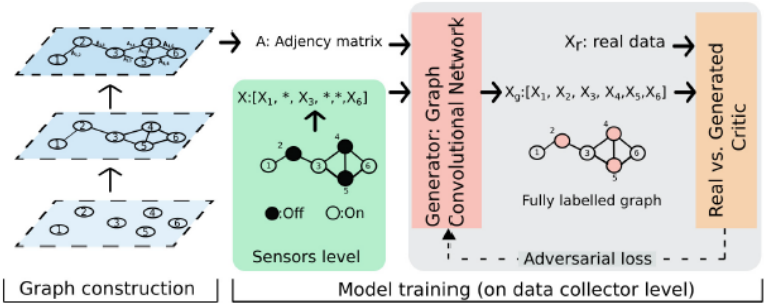

Figure 3.1: General Architecture of GSAVES

The network is modeled as a graph, to capture the spatial relationships between the sensor nodes, and the generator is a graph convolutional network trained adversarially in a semi-supervised manner against a critic network, where the generator replaces the missing data from the sleeping nodes while the critic learns to distinguish between the imputed and the real data. A concrete formulation of the problem would be: Let $X \in 0, 1, *^{N \times d}$ denote the matrix representing sensor data, with N being the sensor network size, and d the number of node attributes representing the length of historical readings. The values 1 and 0 indicate the occurrence or absence of an event, while the symbol * denotes a missing value caused by inactive sensors. The generator receives the adjacency matrix A, representing the mutual coverage between sensors, calculated in the pre-processing phase. A binary mask matrix $M \in 0, 1^{N \times d}$ is defined to express the missing data in X, where each row m_i indicates the presence or absence of reading in x_i. The generator exploits the information in X, M, and the spatial relationships among the sensors described by A to fill in the missing values in X and create a complete matrix $\tilde{Y} \in 0, 1^{N \times d}$ of the generated data. Formally, the generator learns a mapping function f, defined as:

$$\tilde{Y} = f(X, M, A) \tag{3.2.1}$$

The final output $Y \in 0, 1^{N \times d}$ of inputted missing values is then calculated using:

$$Y = M \odot X + \bar{M} \odot \tilde{Y} \tag{3.2.2}$$

where $\bar{M}$ is the logical binary complement of M and $\odot$ denotes the Hadamard product.

The generator is implemented as a Graph Convolutional Network (GCN) that can extract knowledge from graph structures. The critic's role is to challenge the generator during training to produce synthetic data samples closer to the real distribution. Using the W-GAN approach, the goal of the critic is to discover a function c that allows the estimation of the Wasserstein (or Earth-mover) distance between the distribution of the actual data p_r and the distribution of the generated data p_g. The Wasserstein distance is given by:

$$L_c = \sup_c \mathbb{E}x \sim p_r[c(x)] - \mathbb{E}\tilde{x} \sim p_g[c(\tilde{x})] \tag{3.2.3}$$

For performance evaluation, intelligent buildings were chosen as a use-case scenario. They used the MERL dataset, a real dataset collected indoors that includes over 50 million motion sensor events spanning over two years with milliseconds granularity. The experiments compared the proposed solution with four state-of-the-art approaches: GCN, GAIN, GINN, and JGD. They have also evaluated the impact of the sleeping nodes' percentage on data accuracy, varying this percentage from 10The results showed that GSAVES provides the highest accuracy compared to the other solutions, with its F-score slightly improving with the number of nodes. This improvement is attributed to the model's capacity to learn from a larger context, as a higher number of sensors allows the model to better approach global data distribution. For energy performance evaluation, this work considered the average energy consumption per node in terms of milliwatt-second (mWs) while varying the accuracy rates. The energy consumption for a sensor i throughout d time slots was calculated as:

$$E_i = \sum_{j=1}^{d} y_{ij} m_{ij} \; e_{stb}(\sum_{j=1}^{d} m_{ij} - \sum_{j=1}^{d} y_{ij} m_{ij}) \; e_{inact}(d - \sum_{j=1}^{d} m_{ij})] \tag{3.2.4}$$

The experimental results consistently showed that GSAVES guarantees the most extended lifetime among the compared solutions while maintaining the highest accuracy.

3.2.1 Critical Review:

After Carefully examining this work, it was easy to see the innovation in this work compared to other works with the same aim, GSAVES offers a promising solution for the challenge of energy conservation, by combining random sleep scheduling with the data generation, we can clearly see how both the sensing and communication energy consumption are addressed.

However, there are some limitations and areas for improvement: This paper tested the solution on a network of 32 sensors, in IoT networks we have networks with a much higher number of nodes, and it is not clear how GSAVES would perform in such networks. Simi-

larly, the model was only trained on one graph topology assuming a static sensor network, it would have been interesting to test GSAVES on dynamic environments where the relationships between the sensors change overtime. Another important aspect is the security consideration, since the data is generated synthetically, the system is exposed to attacks that manipulate the generator to produce false readings. As for the energy consumption evaluation, a more detailed energy model that considers factors like transmission power, variable sensing rates, or environmental conditions could provide more accurate estimates of energy savings.

Nevertheless, GSAVES is a significant step in energy-efficient IoT sensor networks, the combination of graph neural network with adversarial training is an innovative approach to address energy problems in sensor networks, this work represents a foundation for other future solutions bringing more adjustments and enhancements for a better generalization.

3.3 On Predicting Sensor Readings With Sequence Modeling and Reinforcement Learning for Energy-Efficient IoT Applications:

In this paper, we are introduced to a model that is capable of learning long and short-term spatio-temporal relationships in sensory data and predict future values. This model enables turning sensors off for extended periods to preserve energy. The proposed architecture consists of three main components: LSTM network, RL agent and physical sensors. The LSTM model and physical sensors work together to monitor the network. When the sensors are on, the system relies on their readings. However, the LSTM outputs predictions of the readings when sensors are turned off, On the other hand the RL agent acts as a supervisor here, deciding when to use physical readings or LSTM estimation, balancing power preservation and detection accuracy. Figure 3.2 shows the general framework of the proposed solution in this paper.

3.3.1 LSTM Model for Predicting Sensor Readings:

In this work, an LSTM network is used to learn complex spatiotemporal patterns found in sequential sensor data. The goal is to provide an accurate future event prediction by learning from the past sensor readings using the LSTM's capability of modeling both short-range and long-range dependencies.

This is done first by considering sensor data in pairs of input and output sequences over time (X, Y) . The inputs are the sensor readings and their forecasted values along a timeline. In other words, the LSTM learns iteratively from its previous predictions and

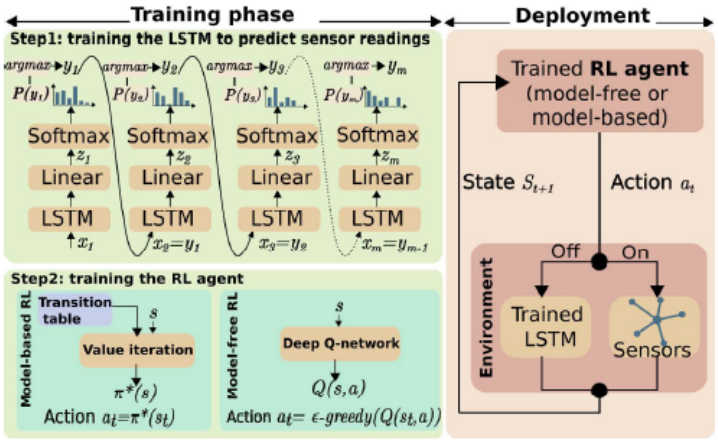

Figure 3.2: General Framework

surroundings to predict the probability of future states, leveraging an extended knowledge of past events for more accurate predictions.

The LSTM's model architecture that lets the network to keep information for some time in its hidden state presents a great advantage. This allows the network to analyse longer sequences of readings, this is particularly useful for tasks involving predicting future sensor states based on historical data. These predictions are made using the softmax layer, which outputs a probability distribution over possible outcomes so that the network can select the most likely future state.

3.3.2 Reinforcement Learning Agent for Optimal Prediction Sequence Length:

In addition to the LSTM, a reinforcement learning (RL) agent is proposed to further optimize the prediction sequence length by determining which sensors should be in the active state. The RL algorithm uses a MDP, where the state is the current readings of the sensors, and the two actions are turning sensors on or off. The RL agent's objective is to maximize prediction accuracy while minimizing energy consumption by selectively controlling the sensors.

The authors propose two approaches to train the RL agent: model-based (LSTM-DP) and model-free (LSTM-DQN). The model-based method uses the value iteration algorithm to compute the optimal policy by evaluating all possible state transitions and actions. But, because of the high computational costs in this method, the paper also propose a model-free solution using Deep Q-Networks (DQN). The DQN approximates the action-value function through a NN, enabling better scalability to larger WSNs.

By combining LSTM for sequence prediction and reinforcement learning for sensor

optimization, the authors effectively address both prediction accuracy and energy efficiency in sensor networks, offering a comprehensive solution for managing large-scale, energy-constrained systems.

3.3.3 Experimental Setup and Results:

Here again, the MERL dataset is used for the model training and evaluation, The experiments focused on three aspects: Prediction accuracy, Energy consumption and scalability. Both model, LSTM-DP and LSTM-DQN were compared to two state of the art solutions: Joint Gaussian Distribution (JGD) and Event Correlations (EC). In addition to that, the authors investigated the impact of learning spatio-temporal correlations by comparing against DNN variants (DNN-DP and DNN-DQN) and a baseline solution (LSTM-IP). The results showed that both LSTM-DP and LSTM-DQN outperformed the state-of-the-art solutions (JGD and EC). The proposed solutions achieved about 50

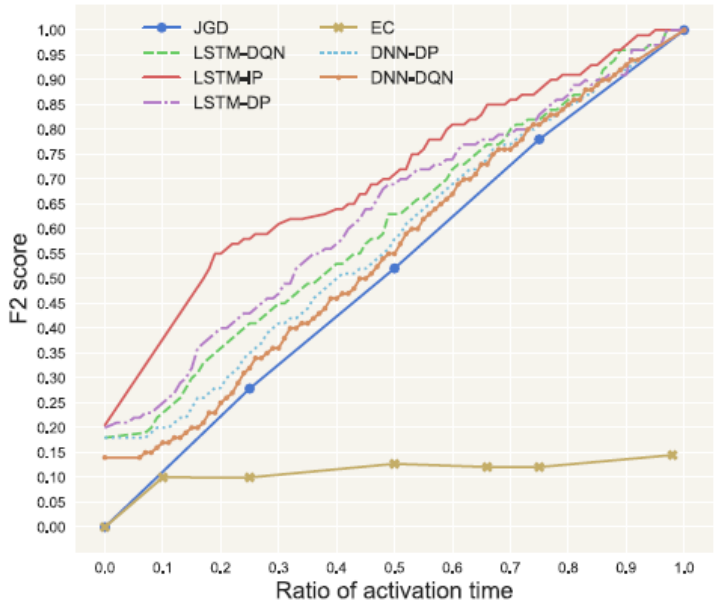

Figure 3.3: F2-score on the test set for four sensors as a function of the ratio of energy.

As for the energy consumption, the experiment consisted of measuring the energy consumed by one sensor while varying the accuracy rate, the final results showed that LSTM-DP achieved better energy savings than LSTM-DQN, and both outperformed JGD.

The scalability of the solution was investigated by varying the number of sensors for 4 to 29, the performance was mesured by: F-score, Latency, Network's lifetime, Average energy consumed by an active sensor, and the ratio of missed events. The experiment shows that extending the network's size increased the F2-score and reduced the ratio of

missed events. It also reduced energy consumption and expanded the lifetime. However, increasing the number of sensors raised the latency.

3.3.4 Critical Review:

This work presents a promising approach for energy-efficient sensor reading prediction in IoT applications. The integration of LSTM for spatiotemporal pattern learning and RL for decision-making is a flexible and effective solution. However, this work also has some limitations that should be addressed to further enhance the results and performance of this solution, like the assumption of a stationary environment, while in a real-case scenario we are often faced with non-stationary environments in IoT networks. The increased latency with the number of sensors is also an issue to be addressed by possibly using parallel processing for time-sensitive applications.

On another point, the scalability of the LSTM-DP model is limited to only 8 sensors, which is probably not useful in real world applications, and for a further generalization of the solution, it would be interesting to evaluate the models on datasets other that the MERL dataset, for different domains that would also allow an extension to a multi-class event detection or continuous value prediction.

Finally, it is important to acknowledge the evaluation that demonstrates significant improvements over existing methods in terms of prediction accuracy and energy consumption.

3.4 PriSTI: A Conditional Diffusion Framework for Spatiotemporal Imputation:

Liu et al. presented in this paper the PriSTI framework, the problem addressed by this framework is relevant in various domains like air quality monitoring, traffic forecasting and other problems that require a spaito-temporal data imputation. The missing data is caused by the sensor failures or data transmission losses, after examining other methods the authors came out with this framework that should address the shortcomings and enhance the data imputation in datasets that have dependencies across both space and time. While the paper does not address directly energy conservation in WSNs, the imputation process proposed here can be accustomed for energy-saving techniques in IoT applications.

3.4.1 Methodology:

PriSTI relies on diffusion probabilistic models (DPMs) for spatiotemporal data imputation, the task at hand is presented as a conditional generation problem. DPMs have shown outstanding success in image composition and audio generation, the authors chose to use them here in a two-stage process: the diffusion process and the reverse process.

Diffusion Process: In this stage, a Gaussian noise is progressively added to the observed spatiotemporal data, making it noisier with time. The goal is to corrupt the data to a point where it become similar to a standard normal distribution. This process can be described as follows:

$$q(X_{1:T}|X_0) = \prod_{t=1}^{T} q(X_t|X_{t-1}), \tag{3.4.1}$$

where $q(X_t|X_{t-1}) = \mathcal{N}(X_t; \sqrt{1-\beta_t}X_{t-1}, \beta_t I)$, with β_t being a small hyperparameter that controls the variance of the added noise. As T increases, the noisy data approaches a standard normal distribution.

Reverse Process: Once the data is noisy enough, the reverse process aims to recover the clean data by iteratively removing the noise. A noise prediction network, informed by the observed data and the geographic relationships among the sensors, guides this process. The reverse process is described as:

$$p_\theta(X_{0:T-1}|X_T, X, A) = \prod_{t=1}^{T} p_\theta(X_{t-1}|X_t, X, A), \tag{3.4.2}$$

where X represents the observed spatiotemporal data and A is the adjacency matrix capturing the geographical information. A conditional feature extraction module is introduced in the paper to capture complex spatiotemporal dependencies. This module processes the noisy data representations, using interpolated data as a global context prior. Then the model predicts noise at each step, weighted by spatial and temporal attention taken from both the data and the geographical informations. The main contributions brought by this paper are, the Conditional diffusion framework that allows the incorporation of the spatial dependencies and the observed values. In the reverse diffusion process a key innovation is observed with the Noise prediction model, It is a crucial part that predicts and removes noise from the corrupted data by using observed data and geographic adjacency, progressively improving the imputed values. Furthermore, PriSTI provides a robust reference point for capturing spatiotemporal patterns by the idea of using interpolated data as a global context.

3.4.2 Training and evaluation:

The training process proposed consists of masking portions of the observed data and training the noise prediction model to learn how to remove the noise, in more detail; we start by masking the data to create the target imputation data. This data is then added to generate conditional information that will guide the model. Finally, Noise is added to the data in several steps, and the model is trained to predict and remove the noise using a loss function that measures the difference between the actual noise and the predicted noise.

After training, the model imputes missing values into the noisy imputation target sampled from a normal distribution, Then, The model gradually removes the noise in a reverse process, recovering the missing data by sampling values from predicted distributions at each time step.

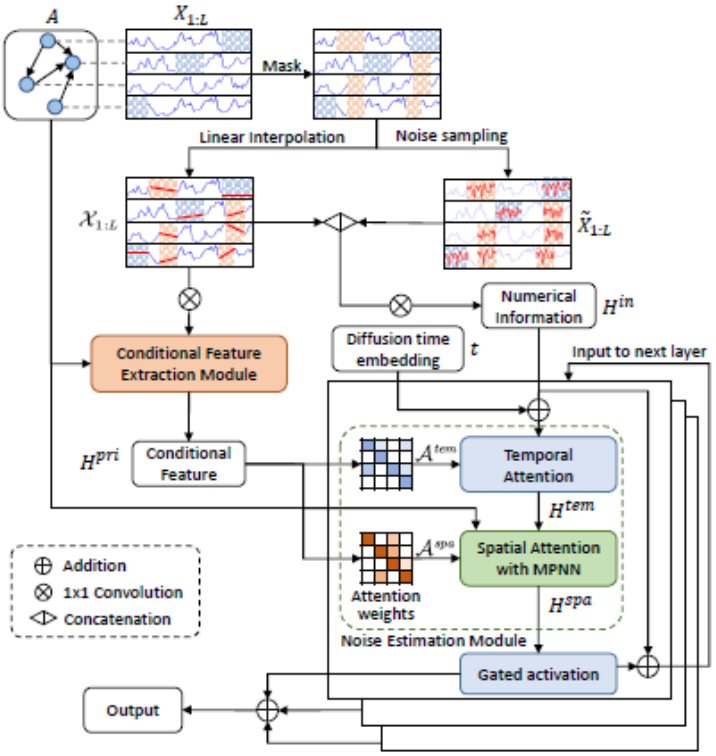

Figure 3.4: The pipeline of PriSTI.

The paper outlined 3 real world datasets that were used for the model evaluation: AQI-36 (air quality data), METR-LA, and PEMS-BAY (traffic data). The results show that PriSTI outperforms several baseline methods, including traditional statistical models, machine learning approaches, and other deep learning-based imputation models, across various missing data patterns, It is easily shown that PriSTI excels in scenarios with high missing rates and sensor failures, where other models struggle. The evaluation was carried

out using metrics such as MAE and CRPS. For example, on the AQI-36 dataset, PriSTI achieved a MAE of 9.03 for simulated failure patterns, which outperforms the CSDI model, which had a MAE of 10.56. On the other hand, on the PEMS-BAY dataset, PriSTI achieved a CRPS of 0.0064, while CSDI scored 0.0067.

3.4.3 Critical Review:

This paper brought an innovative approach by using generative AI for spatiotempral data imputation, it also introduced various key algorithmic additions, addressing the spatial relationships in sensor networks, an aspect which is often neglected in other studies.

However, sine PriSTI depends on Gaussian noise processes and interpolation-based conditional information some limitations are introduced. First, while interpolation is efficient, it does not always capture the complex non-linear patterns present in real-world data, especially for datasets with highly dynamic spatiotemporal relationships. Additionally, the computational complexity of PriSTI, particularly in terms of training and inference time, is higher than some existing methods, making it less practical for large-scale real-time applications, and impractical for sensor networks due to their low computational capacity.

Finally, while PriSTI have good results in scenarios with high missing rates, its performance on datasets with low missing rates does not really outperform other methods, so we can say that the strengths of this framework are most useful in particularly challenging imputation tasks.

3.5 TARNet: Task-Aware Reconstruction for Time-Series Transformer:

In this paper, Ranak Roy Chowdhury et al. introduce TARNet, an approach to improve time series classification and regression tasks that relies on task-aware reconstruction. This innovative approach is used to address issues faced by the other state of art methods in deep learning that struggle with semi labeled data.

The authors found that unsupervised pre-training through reconstruction was explored as a way to leverage unlabeled data, but methods like Time Series Transformer (TST) use random masking, which may ignore that certain timestamps are more important for a certain task. For which, they introduced task-aware reconstruction, where the masking strategy is guided by information from the end task.

While this paper does not address the energy conservation task specifically, it is a novel approach for time series classification which can be explored for energy saving in WSNs.

3.5.1 Methodology:

Ranak Roy Chowdhury et al. start with a multivariate time series $X \in \mathbb{R}^{S \times N}$ with S timesteps and N variables, along with a target label y, the goal of TARnet is to predict labels $\tilde{y}$ for unseen data X.

The core of the model is a transformer encoder used starting with a mean standardization of input features, then the feature vectors are linearly projected, followed by an addition of positional encoding then, the processing is done in transformer encoder blocks, finally the weighted outputs are passe through a feed-forward network.

The architecture of TARnet consists of three components: 1. End Task (T_{END}): A standard supervised task used for classification and regression, the model uses the vector from the last timestamp with two fully connected layers with ReLU activation and finally, An output layer.

For classification, the authors applied a softmax to obtain class probabilities, with a Cross-entropy loss for classification and a Squared error loss for regression.

Task-aware Reconstruction: This is an unsupervised reconstruction task with task-informed masking, it aims to recover the input data X after masking. the most important used notions are:

1. A binary mask $m \in \mathbb{R}^S$ generated by the masking strategy M.
2. Masking involves replacing feature vectors with zeros at selected timestamps.
3. The masked input is processed through Transformer Encoder layers.
4. The output is passed through fully-connected layer to get the reconstructed data $\tilde{X}$.

The reconstruction loss L_{TAR} is a weighted sum of masked and unmasked losses, and the total loss is a combination of the end task and reconstruction losses.

Data-driven Masking Strategy (M): This mechanism is used to select important timestamps for masking based on final task characteristics, this is the one of the innovation brought by this paper, this strategy uses self-attention weights from the end task to identify important timestamps, the proposed process goes like:

1. Computing aggregate attention map $A \in \mathbb{R}^{S \times S}$ from Transformer Encoder layers.
2. Calculating normalized aggregate attention weights $\sigma \in \mathbb{R}^S$: $\sigma_k = \frac{\sum_{i=1}^{S} A_{ik}}{\sum_{k=1}^{S} \sum_{i=1}^{S} A_{ik}}$
3. Selecting top $\lfloor \beta S \rfloor$ values from σ to form σ'.
4. Randomly sampling $\lfloor \mu S \rfloor$ timestamps without replacement from σ' to generate m.

This approach ensures that important timestamps for the end task are masked and reconstructed with Randomization added to prevent overfitting to a fixed set of timestamps.

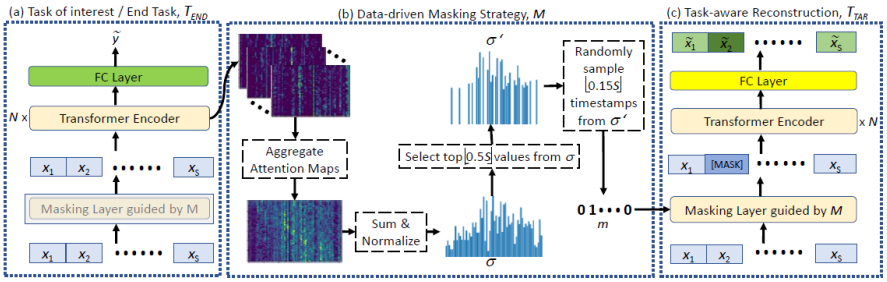

Figure 3.5: Overview of TARnet.

3.5.2 Training and evaluation:

The authors evaluate TARNet on a diverse set of time series classification and regression tasks, comparing it against numerous statistical and deep learning baselines, unsing numerous data benchmarks covering various domains (e.g., motion, audio, EEG, HAR) from UEA Archive, UCI Machine Learning Repository, Monash University, UEA, UCR Time Series Regression Archive.

The complete training procedure for TARNet is shown in Algorithm 1

Algorithm 1 Training of TARNet

Input: X, y
Hyper-parameters: μ, β, λ, η
Output: *Model*

σ initialized randomly
Model = TransformerEncoder()
while training **do**
  σ' = top $\lfloor \beta S \rfloor$ values from σ
  $m \sim$ Randomly sample $\lfloor \mu S \rfloor$ timestamps without replacement from σ'
  $\tilde{X}, \tilde{y}$, A = *Model*.train(X, m) # A ← Self-Attention Scores
  Compute $\mathcal{L}_{TAR}(\tilde{X}, X, \lambda)$ and $\mathcal{L}_{END}(\tilde{y}, y)$
  $\mathcal{L}_{Total} = \eta \mathcal{L}_{TAR} + (1 - \eta)\mathcal{L}_{END}$
  σ = add_and_normalize(A)
end while
return *Model*

Figure 3.6: TARnet training.

For the sake of evaluation, the authors compared TARnet to numerous methods and baselines, including Time Series Transformer (TST), TS2Vec, TNC, TS-TCC, ResNet, ShapeNet and WEASEL-MUSE, Using various metrics for a fair evaluation including: 1. Accuracy (for classification) and RMSE (for regression) 2. Ours 1-to-1 Wins/Draws/Losses: Number of datasets where TARNet performs better/same/worse than baselines. 3. Mean Rank: Average rank of each model across datasets. 4. Avg.Rel.Diff.Mean: Average relative difference from mean performance across datasets

The results of the comparison TARnet show superiority to other methods, with the

highest average accuracy (77.2 Similarly TARnet showed great results for regression tasks,

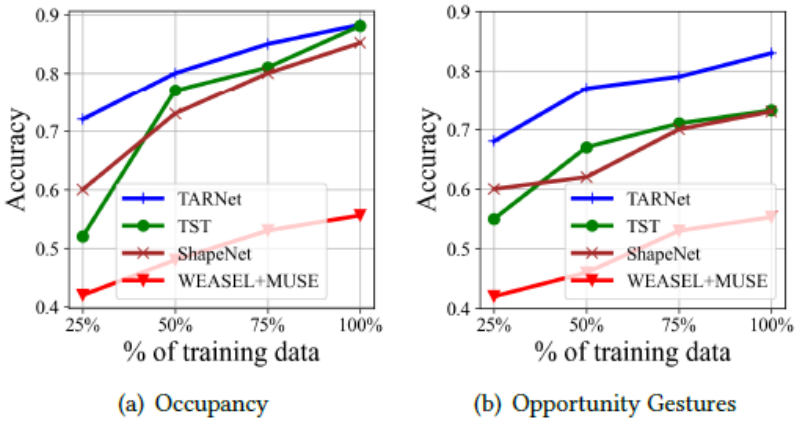

Figure 3.7: TARnet classification results.

ranking first on three datasets and second on two datasets, outperforming all baseline models, with the lowest mean rank (1.833) across all datasets, compared to 2.5 for the next best model (TST), and a 31.3

3.5.3 Critical Review:

After reviewing this paper, it is safe to say that TARnet presents a novel and promising approach to improving time series analysis tasks, the authors addressed previous problems in models performing the same task as TARnet and they delivered a comprehensive evaluation showing how their model outperforms other methods, that is die to their Innovative masking strategy and the flexibility of the architecture and an improved data efficiency.

However, this work shows some limitation due to the potentially very high computational complexity of the model compared to the other baseline models, which makes this approach highly inappropriate for energy conservation in IoT environments and especially for WSNs, also, this method introduced many other new hyperparameters which may make the highly model sensitive leading to lower performance in simpler environments like WSNs, additionally , a further generalization of this method is needed in other architectures to see how task-aware reconstruction performs.

In conclusion, TARNet represents a significant contribution to the field of time series analysis, introducing a novel approach that combines supervised and unsupervised learning in a task-aware manner, however due to the limitations that we already sited, this method is not suitable for energy aware conservation tasks in WSNs.

we can summarize the key findings from each model in table 3.1:

Method	Strengths	Limits
Reinforcement Learning for WSNs (SARSA)	- Adaptive to changing environments - Balances exploration and exploitation - Can optimize for multiple objectives - Learns from experience without explicit programming	- Can be slow to converge in large state spaces - May require significant training data - Potential for unstable behavior during learning - Computational complexity can be high for resource-constrained devices
GSAVES (Graph Neural Networks)	- Captures spatial relationships between sensors - Can generate missing data, allowing more sensors to sleep - Adversarial training improves data generation quality - Works well with event-based data	- struggles with highly dynamic network topologies - Requires centralized processing. - Performance may degrade with very sparse data - Training can be computationally intensive
LSTM for Sensor Reading Prediction	- Captures both short-term and long-term temporal dependencies - Can handle multivariate time series data - Provides accurate predictions for future sensor states - Allows for longer sensor sleep periods	- High memory requirements - Can be computationally expensive for training and inference - May struggle with very long sequences - Requires careful tuning of hyperparameters
Diffusion Probabilistic Models (PriSTI)	- Excellent for handling missing data and imputation - Can capture complex spatiotemporal dependencies - Robust to noise and outliers - Generates high-quality synthetic data	- Very computationally intensive, especially for training - May be too complex for simple WSN tasks - Requires significant data for training - Can be slow for real-time applications in WSNs
Transformers for Time Series (TARNet)	- Excellent at capturing long-range dependencies - Parallelizable, potentially faster than RNNs - Can handle variable-length sequences - Attention mechanism provides interpretability	- High memory requirements, especially for long sequences - May struggle with very fine-grained temporal patterns - Requires large amounts of data for effective training - Computationally intensive, may be overkill for simple WSN tasks

Table 3.1: Comparative Table

3.6 Synthesis:

After Reviewing the current literature on energy conservation and data imputation in wireless sensor networks and the Internet of Things applications, we notice that the field has seen a shift towards the newest machine learning and deep learning techniques to address communication and activation costs on energy while trying to keep the quality of data in the network, we can categorize this problem into two main directions:

Graph Based Methods: Techniques like GSAVES where GCN in used to capture the spatial relationships between sensors in the network, techniques like this show potential in handling the interconnected nature of sensor networks and allowing for selective sensor deactivation while generating the data, which keeps the data integrity, the usage of GAN was also innovative since it allowed for more sensors to enter the sleeping mode without significant loss of information. However, Graph-based methods are hardly operable with highly dynamic network topologies and sparse data. Another major challenge is that most of the graph-based methods require central processing, which can be very burdensome for large-scale deployments.

Time Series Methods: Models in this class, including but not limited to LSTM-based prediction models, employ a transformer-based architecture similar to TARNet that learns temporal dependencies from sensor data. These techniques are particularly good for predicting future sensor states, which can potentially allow longer sleep periods and, therefore, more power-efficient duty cycling.The LSTM-based approach, combined with reinforcement learning, shows the most promise in balancing between the prediction accuracy and energy conservation. Meanwhile, transformer-based models, such as TARNet, introduce innovative concepts, like task-aware reconstruction, which could be adapted for energy-efficient sensing tasks. However, most of these time series methods require high computational and memory needs, which resource-constrained WSN devices cannot provide. They may also perform poorly with very long sequences or might expect too much training data.

We can also notice that most of the proposed methods are tested on small to medium-scale networks; therefore, it is still challenging to investigate how they adapt to large-scale IoT deployments, and while a few methods exist, such as reinforcement learning methods, which can adapt to changing environments, many still assume relatively static conditions of the network. And while some methods showcase promising results in terms of accuracy of prediction or data imputation, the complexity of more these models is often high for resource constrained WSNs, possibly offsetting energy savings.

In this scope, and after acknowledging the limitations of previous efforts, we can direct future research into more hybrid approaches, for instance, integration graph-based methods with advanced techniques in time series analysis to result in improved models representing both the spatial and temporal aspects of WSNs, we can also employ Federated Learning for WSNs, where The study of federated learning approaches would enable the sharing of model training across distributed sensors without data centralization, aiming at an improved privacy-energy efficiency trade-off. Furthermore, Transfer learning can probably allow models trained on resource-rich environments to be effectively used on resource-constrained ones with a minimum need to do extensive on-device training.

Conclusion and Future Work

From environmental monitoring to industrial automation, WSNs have emerged as an important tool in many applications. Energy efficiency remains a major concern due to the limited battery life of sensor nodes. This thesis presents several solutions to address these energy limitations, focusing on traditional methods like duty cycling and new ML trends.

Conventionally, duty cycling is used to reduce energy consumption by switching sensor nodes between active and sleep states. However, this can create data gaps, requiring methods to accurately fill in the missing data. We also explored other methods like Data aggregation and Energy Harvesting but despite their effectiveness, traditional approaches are often dependent on static network conditions and lack flexibility in dynamic environments.

More advanced methods, as discussed in the literature, involve RL and sequence modeling. RL techniques like SARSA and MDP dynamically optimize duty-cycling parameters with positive results. Likewise, LSTM models have excelled at predicting future sensor states based on past data, allowing for longer sleep cycles without sacrificing data integrity.

Despite significant progress, challenges remain in applying these methods to real-world situations. Future work should focus on making these algorithms more robust against changing network conditions and sensor failures. Hybrid approaches that combine spatial modeling with advanced time-series techniques offer additional energy savings. Furthermore, federated learning and graph convolutional networks represent promising directions for decentralized model training and spatial data representation.

Thus, future research based on the reviewed methods can help develop more energy-efficient, adaptive WSN systems capable of functioning in dynamic and constrained environments.

Bibliography

[1] R. R. Chowdhury, X. Zhang, J. Shang, R. K. Gupta, and D. Hong, *Tarnet: Task-aware reconstruction for time-series transformer*, in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 212–220, 2022.

[2] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, *Machine learning in wireless sensor networks: Algorithms, strategies, and applications*, *IEEE Communications Surveys & Tutorials* **16** (2014), no. 4 1996–2018.

[3] Z. Li, K. Zhang, Y. Zhang, Y. Liu, and Y. Chen, *D2d-assisted adaptive federated learning in energy-constrained edge computing*, *Applied Sciences* **14** (2024), no. 12 4989.

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, *A comprehensive survey on graph neural networks*, *IEEE transactions on neural networks and learning systems* **32** (2020), no. 1 4–24.

[5] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, *Machine learning paradigms for next-generation wireless networks*, *IEEE Wireless Communications* **24** (2016), no. 2 98–105.

[6] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, *Transformers in time series: A survey*, *arXiv preprint arXiv:2202.07125* (2022).

[7] C. R. Wren, Y. A. Ivanov, D. Leigh, and J. Westhues, *The merl motion detector dataset*, in *Proceedings of the 2007 workshop on Massive datasets*, pp. 10–14, 2007.

[8] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, *Informer: Beyond efficient transformer for long sequence time-series forecasting*, in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, pp. 11106–11115, 2021.

[9] J. Kim, D. Nguyen, S. Min, S. Cho, M. Lee, H. Lee, and S. Hong, *Pure transformers are powerful graph learners*, *Advances in Neural Information Processing Systems* **35** (2022) 14582–14595.

[10] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, *Diffusion models: A comprehensive survey of methods and applications*, *ACM Computing Surveys* **56** (2023), no. 4 1–39.

[11] V. P. Dwivedi and X. Bresson, *A generalization of transformer networks to graphs*, *arXiv preprint arXiv:2012.09699* (2020).

[12] J. Chen, X. Cao, P. Cheng, Y. Xiao, and Y. Sun, *Distributed collaborative control for industrial automation with wireless sensor and actuator networks*, *IEEE Transactions on Industrial Electronics* **57** (2010), no. 12 4219–4230.

[13] M. Kozlowski, R. McConville, R. Santos-Rodriguez, and R. Piechocki, *Energy efficiency in reinforcement learning for wireless sensor networks*, *arXiv preprint arXiv:1812.02538* (2018).

[14] M. Liu, H. Huang, H. Feng, L. Sun, B. Du, and Y. Fu, *Pristi: A conditional diffusion framework for spatiotemporal imputation*, in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pp. 1927–1939, IEEE, 2023.

[15] C. Meijer and L. Y. Chen, *The rise of diffusion models in time-series forecasting*, *arXiv preprint arXiv:2401.03006* (2024).

[16] R. Laidi, D. Djenouri, and I. Balasingham, *On predicting sensor readings with sequence modeling and reinforcement learning for energy-efficient iot applications*, *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **52** (2021), no. 8 5140–5151.

[17] R. Laidi, D. Djenouri, M. Bagaa, L. Khelladi, and Y. Djenouri, *Generating event sensor readings using spatial correlations and a graph sensor adversarial model for energy saving in iot: Gsaves*, in *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, IEEE, 2023.

[18] W. L. Hamilton, *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[19] S. Suresh, P. Li, C. Hao, and J. Neville, *Adversarial graph augmentation to improve graph contrastive learning*, *Advances in Neural Information Processing Systems* **34** (2021) 15920–15933.

[20] L. Lin, Z. Li, R. Li, X. Li, and J. Gao, *Diffusion models for time-series applications: a survey*, *Frontiers of Information Technology & Electronic Engineering* **25** (2024), no. 1 19–41.

[21] A. Behrouz and F. Hashemi, *Graph mamba: Towards learning on graphs with state space models*, in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 119–130, 2024.

[22] S. Goel and T. Imielinski, *Prediction-based monitoring in sensor networks: taking lessons from mpeg*, *ACM SIGCOMM Computer Communication Review* **31** (2001), no. 5 82–98.

[23] R. C. Carrano, D. Passos, L. C. Magalhaes, and C. V. Albuquerque, *Survey and taxonomy of duty cycling mechanisms in wireless sensor networks*, *IEEE Communications Surveys & Tutorials* **16** (2013), no. 1 181–194.

[24] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, *Energy-efficient communication protocol for wireless microsensor networks*, in *Proceedings of the 33rd annual Hawaii international conference on system sciences*, pp. 10–pp, IEEE, 2000.

[25] A. Imteaj, K. Mamun Ahmed, U. Thakker, S. Wang, J. Li, and M. H. Amini, *Federated learning for resource-constrained iot devices: Panoramas and state of the art*, *Federated and Transfer Learning* (2022) 7–27.

[26] X. Zheng, Y. Wang, Y. Liu, M. Li, M. Zhang, D. Jin, P. S. Yu, and S. Pan, *Graph neural networks for graphs with heterophily: A survey*, *arXiv preprint arXiv:2202.07082* (2022).

[27] A. Jain and E. Y. Chang, *Adaptive sampling for sensor networks*, in *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*, pp. 10–16, 2004.

[28] C. Caione, D. Brunelli, and L. Benini, *Distributed compressive sampling for lifetime optimization in dense wireless sensor networks*, *IEEE Transactions on Industrial Informatics* **8** (2011), no. 1 30–40.