Machine Learning Approaches for Anomaly Detection and Load Forecasting in Smart Buildings Time Series Data

Maher Dissem

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

December 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Maher Dissem**

Entitled: **Machine Learning Approaches for Anomaly Detection and Load Forecasting in Smart Buildings Time Series Data**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________ Chair and examiner
Dr. Chun Wang

______________________ Examiner
Dr. Arash Mohammadi

______________________ Supervisor
Dr. Manar Amayri

Approved by ______________________
Chun Wang, Chair
Department of Concordia Institute for Information Systems Engineering

__________ 2024

____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Machine Learning Approaches for Anomaly Detection and Load Forecasting in Smart Buildings Time Series Data

Maher Dissem

The proliferation of IoT sensors in smart buildings has enabled extensive time-series data collection, providing valuable insights for predictive maintenance, operational efficiency, and energy management. This data can be used within an anomaly detection framework to automatically identify abnormal behaviors, enabling the detection of issues related to power consumption, control system failures, and sensor malfunctions. Additionally, it supports load forecasting, crucial for optimizing energy management by enabling efficient resource allocation, cost savings, and sustainable operations through accurate demand prediction. However, IoT devices are prone to issues like hardware malfunctions and transmission errors, which introduce data anomalies and complicate the effective use of collected data. A popular and effective anomaly detection framework trains autoencoders on minimizing the error between original and reconstructed sequences. By setting a threshold on the reconstruction error, abnormal sequences can be distinguished from the predominant regular patterns. However, this method is highly sensitive to architectural parameters and the nature of anomalies, making it difficult to develop a universally effective method or fine-tune a model without prior knowledge of the building and sensors settings. To address this, we propose a reinforcement learning-based neural architecture search approach to explore a manually defined search space and identify the optimal neural configuration through trial and error. While this method demonstrates competitive performance in discovering effective architectures that may be not intuitive, it assumes the availability of anomaly-free data for training, which is not practical in real-world scenarios. Hence, we also present an unsupervised feature bank-based model for anomaly detection in anomaly-contaminated time series. We integrate this with a recurrent denoising autoencoder, trained on data deemed anomaly-free, to replace identified anomalies with plausible patterns. This results in a combined anomaly detection and imputation pipeline that preprocesses data for downstream tasks, such as forecasting, in which we compare the performance of different models before and after preprocessing and demonstrate significant improvements. Despite these advancements, data scarcity remains a challenge in training effective load forecasting models for smart buildings, specially that concerns over the privacy of IoT sensor data make building owners reluctant to share their data, whether for transfer learning or for centralized model training, as it can reveal sensitive information about the occupants' behaviors. To overcome this limitation, we introduce a federated load forecasting framework to exploit data from multiple buildings while maintaining data privacy. Acknowledging the diverse load profiles shaped by factors like size, location, and user behavior, we also investigate existing techniques to personalize the global model to accommodate each building's specific characteristics.

Acknowledgments

I would like to express my heartfelt gratitude to my research supervisor, Dr. Manar Amayri, for her invaluable guidance and mentorship throughout my degree. Her insightful feedback has significantly enhanced the quality of this work, and her commitment and support have made this journey both fulfilling and enjoyable.

I am also deeply grateful to Prof. Nizar Bouguila for his valuable input and stimulating discussions. His expertise and perspectives have enriched my understanding of various topics explored in this research.

My appreciation extends to the dedicated faculty at Concordia University, whose inspiring lectures and academic rigor have profoundly influenced my intellectual growth.

I am also grateful to my fellow lab members for their camaraderie and collaborative spirit, which have made our work environment both stimulating and supportive.

Finally, I would like to acknowledge the unwavering support of my family and friends. Their love and encouragement have been instrumental throughout my pursuit of this academic endeavor.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In the modern era, buildings are undergoing a remarkable transformation, evolving into intelligent and responsive entities thanks to the integration of IoT (Internet of Things) sensors. These sensors, deployed in various areas of buildings, enable them to gather, process, and analyze vast amounts of data from their environment that has the potential to offer valuable insights when properly leveraged [58].

The main drivers for the adoption of these sensors is to enable better management of energy, ultimately leading to energy and cost saving and to comply with the legislations that have recently been set by various governments [4] to optimize energy use by encouraging the deployment of smart meters and intelligent control systems for various building components, such as Heating, Ventilation and Air Conditioning (HVAC) systems, lighting, pumps, lifts, etc.

This matter is particularly pressing as the global energy consumption continues to rise [57, 65], with the building sector accounting for a significant portion of this demand. According to recent estimates, buildings are responsible for approximately 35% to 40% of total global energy consumption [97, 66]. Therefore, enhancing buildings' energy efficiency is an effective approach to save considerable amounts of energy, to significantly reduce greenhouse gas emissions [66] and overall to help mitigate climate change.

Multiples techniques have been proposed over time to prevent energy waste in buildings such as thermal insulation, efficient glazing solutions, optimized spatial planning and natural ventilation strategies [40] or more energy efficient technologies for motors, pumps and fans [1]. In particular, incorporating IoT sensors and their smart applications results in around 30-50% energy savings in comparison with traditional buildings [112]. These applications include data-driven automated decision-making and the use of Machine Learning (ML) techniques to optimize various aspects of buildings such as maintaining comfortable indoor environments through the autonomous adjustment of HVAC systems to have it operate on partial load according to co_2 levels, temperature, humidity and occupancy data. For example, real-time occupancy data gathered from various sensors can be used to optimize the systems used for transporting people, such as escalators and elevators which account for 2-5% of the total energy consumption of large commercial buildings or to adjust water circulation depending on the need and stop water pumps during silent hours [1]. Additionally, for domestic buildings, it was shown that providing real-time meter readings to tenants as feedback on their energy consumption, incite them to adjust their behavior leading to energy saving in the range of 5-15% [31].

Accurate load forecasting, achieved by harnessing historical data and predictive analytics, is one of the most important application of IoT sensor data as it plays a key role in efficient resource allocation, cost reduction, and sustainable operations. In fact, it contributes to estimating a building's occupancy level, thereby assisting in devising optimal control strategies for smart equipment such as lighting and HVAC systems [68]. Optimizing the energy consumption of HVAC systems is particularly important for smart buildings, as they account for 50% of a building's energy use [1]. It also allows the reduction of energy waste by anticipating peak demand periods and allowing energy to be produced, stored, or redirected, thereby alleviating grid strain and lowering operational costs [18]. In chiller plants, for instance, operations can be optimized based on anticipated cooling loads, resulting in notable energy savings [116]. Additionally, these forecasts can detect emerging inefficiencies in electrical systems, signaling when maintenance is required by identifying gradual or abrupt performance signaling the need for maintenance [33]. At the individual level, forecasting appliance-level energy consumption has been shown to positively influence consumer behavior, improving their behaviors towards energy savings [62]. For facility managers, it enables sizing and managing micro-grid systems in order to cover demand fluctuation while accounting for power production and energy storage level [50]. For electricity suppliers, it allows energy suppliers to optimize electricity production [12] and distribution strategies [22] to meet consumers requirements while improving energy efficiency and reducing greenhouse gas emissions. Furthermore, it enables governments to anticipate future energy needs, empowering them to make well-informed decisions in resource and infrastructure planning such as the allocation of renewable energy sources, power plants, and storage systems to ensure power grids' capacity meets demand at the regional level [53].

1.2 Problem Statement

While the deployment of IoT sensors in smart buildings offers promising advancements in fault detection and energy management, these sensors are prone to recording time series anomalies due to issues such as hardware malfunctions, communication failures, and power disruptions. Such anomalies can disrupt the performance of machine learning models trained on this data. For Deep Learning (DL), for example, they may lead to abnormal loss values during training, resulting in suboptimal gradient updates in neural networks. Effective anomaly detection is therefore necessary, not only to automatically signal these issues and prompt maintenance or further investigation, but also to preprocess historical data before using it for downstream tasks like occupancy estimation or load forecasting.

The complexity of anomaly detection is aggravated by the significant diversity in load profiles across different buildings, influenced by factors like size, location, climate, sensor types, occupant habits, etc. Consequently, anomalies vary from one sensor and one building to another, posing a challenge for traditional detection frameworks like reconstruction autoencoders as these models are highly sensitive to parameter settings, and small adjustments can significantly alter their detection capabilities, potentially leaving some anomalies undetected. As a result, creating a universal anomaly detection model that adapts well to diverse building characteristics is not possible, and tailoring one for every building requires extensive manual model fine-tuning along with prior knowledge of the building and sensors configurations.

A related challenge lies in obtaining the clean, anomaly-free training data, required for training unsupervised anomaly detection models that rely on identifying deviations from typical patterns.

This is because manually identifying anomalies to filter them from the training set of unsupervised methods or to create reliable labels for supervised classifiers is not only time-consuming and costly but also prone to errors. Furthermore, while for classification tasks like occupancy estimation, the detected anomalies can be omitted from the training data, for load forecasting, we would have to discard long sequences as both the model's input and forecast horizon should be anomaly free, hence significantly decreasing the size of the training dataset. Therefore, it's better to impute these anomalies with plausible values to maintain data integrity for these downstream tasks.

Moreover, while anomaly detection and imputation can enhance the performance of load forecasting deep learning models, a challenge remains for buildings with limited historical data, such as newly constructed or repurposed structures. In such cases, the training of accurate ML models is restricted by the insufficient amount of data available for training. Such a problem is typically addressed by either collecting raw data from similar buildings for training or through Transfer Learning (TL), where a model trained on a given building is adapted to another building. However, in the second case, it is necessary to analyze the datasets for similarity to ensure that the source and target datasets are sufficiently aligned for effective knowledge transfer. However, privacy concerns prevent building owners and occupants from sharing IoT data, as they could reveal sensitive information about their habits and behaviors [104]. Federated Learning (FL) offers a potential solution by enabling multiple buildings to collaboratively train a model without the need for data sharing, but the previously mentioned inherent data heterogeneity across buildings may reduce the effectiveness of FL frameworks, limiting their ability to generalize across diverse building profiles.

1.3 Literature Review

In this section, we review the state-of-the-art literature on anomaly detection methods, prevalent imputation techniques, and established time series forecasting models. We also discuss recent advancements in automating the optimization of machine learning models and explore strategies for federated learning, including personalization approaches.

1.3.1 Time Series Anomaly Detection

Time series anomaly detection has evolved significantly from traditional methods, such as statistical models relying on measures like mean, variance, and median or time and frequency domain based signal analysis methods such as employing Fourier Analysis [26]. In fact, these conventional approaches have been surpassed by machine learning techniques, notably supervised machine learning models such as classification algorithms [98]. For instance, in [139], an approach was proposed to combine autoencoder denoising, and feedforward neural networks, hence establishing a multi-layer hierarchical network capable of detecting anomalies in single household energy consumption data and identifying the household appliance in question through classification. The approach used elaborate feature engineering to exploit time seasonality. However, these supervised methods require significant amounts of labelled data and the diverse nature of anomalies, spanning subtle to significant deviations, make it challenging and costly to correctly identify and label anomalies within large datasets in the smart buildings' context.

Weakly supervised and semi-supervised approaches were therefore proposed, such as [144] which introduces an optimization algorithm for partially labeled time series datasets or [54] where authors develop a neural network based electricity fraud detection system for users' consumption profiles and train it in a semi-supervised way through alternate multitask learning.

However, ideally, an anomaly detection model should operate without the need for labeled data. A promising approach involves gathering datasets comprising solely normal samples, framing the problem as an out-of-distribution detection task where a model is trained on normal data only and should be able to detect at test-time samples that do not belong to the training data distribution.

This unsupervised, out-of-distribution detection paradigm may be realized through clustering methods that map time-series data into a multidimensional space, then classify newly received data samples as anomalies if they are far from pre-defined clusters or have low probability of belonging in any of the clusters such as [70] that exploits the Fuzzy C-Means clustering algorithm to detect anomalies in multivariate time series data or [6], where authors used data autocorrelation and clustering methods to identify data anomaly in household appliances. Another attempt for anomaly detection using unsupervised machine learning is presented in [10]. This work presents a system for monitoring the energy consumption of buildings and detecting anomalies by assigning an anomaly score to a single user according to their usage history and then combining data from multiple users and taking advantage of clustering models.

This task can also be framed as a distance-based task, where the explicit distance between two temporal sequences quantifies their similarity. Anomalies are identified when newly obtained sequences deviate significantly from the expected range of distances from normal sequences.

Furthermore, this concept can be extended to unsupervised deep learning settings using reconstruction models, where the goal is to learn inherent representations of normal data. In this scenario, models that have not encountered anomalies during training exhibit distinct behavior during inference. Abnormal samples are detected using a distance measure, such as leveraging the learned hidden representations from a reconstruction autoencoder or quantifying the disparity between input data and the reconstruction of an anomalous sample. This discrepancy arises because the model's knowledge is based on patterns learned from clean data, as discussed in [144].

In this context, [95] employs a pretrained specialized feature extraction neural network to encode image samples and perform this comparison in a new rich feature space where regular and anomalous features are easier to distinguish. Their approach relies on constructing a feature memory bank, assuming the embeddings computed for anomalous test-time time series will be vastly different.

In a related vein, [2] proposes the use of a multivariate Long Short-Term Memory (LSTM) based autoencoder for anomaly and event detection in ambient $PM_{2.5}$, temperature, and humidity measurements captured by IoT devices and proposes to optimize the device's battery consumption by dynamically changing the detection period using a game theory inspired approach. [130] uses a two-stage sliding window and a Convolution Neural Networks (CNN) for feature extraction combined with a LSTM based autoencoder and a deep neural network for anomaly detection in IoT time series data. Furthermore, in [138], the authors propose a multi-variate time series anomaly detection framework based on the reconstruction of a feature matrix using a convolutional autoencoder and test it on a dataset of house monitoring and emergency detection. In addition, graph neural network (GNN) based models offer another avenue for anomaly detection. In [107], for instance, authors propose to learn the normal patterns of multivariate time series and utilize reconstruction probability for anomaly detection. Also, [82] proposes a Federated Learning-based anomaly detection framework for industrial IoT, where authors combine attention-based CNN units with a LSTM network to detect anomalies in power demand time series data. Similarly, [88] makes use of a CNN and a two-stage LSTM-based autoencoder to detect anomalies and rare events hidden in sensor data of Industrial IoT.

Additionally, efforts have been made to develop novel anomaly detection frameworks that take advantage of Reinforcement Learning, an ML paradigm where agents learn to make decisions by interacting with an environment and receiving feedback on their actions in terms of rewards or penalties according to the action's outcome. The agent aims to maximize its cumulative reward over time by discovering optimal policies through trial and error [11].

For instance, in [55], authors leverage RL to avoid the intricate task of threshold tuning and propose a technique to detect anomalies in time series data without assuming the underlying mechanisms responsible for anomalies. They use the standard Q learning framework with an LSTM based agent that decides whether a portion of time series data contains an anomaly or not. [126] addresses the lack of labelled data by developing a framework based on Deep Q Learning and incorporating Active Learning and Label Propagation techniques to achieve competitive anomaly detection performance while only manually labelling a tiny fraction of samples for training. [125] addresses the task of detecting anomalies in the data of smart buildings sensors. They generate their own dataset by injecting three types of noise into regular sensor measurements and apply a Deep Reinforcement Learning approach to perform the hyperparameter optimization of SBS [45], a building specific anomaly detection algorithm.

Besides this, in the context of smart buildings, RL is mostly used for control application, such as finding an optimal control strategy for HVAC [46, 13, 86] or for modern lighting, blinds and window systems [35] to achieve reasonable occupant comfort level while reducing energy consumption.

However, all of these methods face a major challenge: they don't work well when training data contains anomalies. In real-world situations, it's unrealistic to expect completely anomaly-free training data. This limitation must be addressed for reliable anomaly detection in practical applications.

To address this issue, an alternative approach, proposed in [59], enhances the previously mentioned memory-bank technique [95] by storing the extracted image features in a memory bank and subsequently denoising it. This method enables learning from noisy and anomaly-infected data. However, experiments were only conducted for image datasets, and it is not clear how it may perform for univariate time series data.

1.3.2 Time Series Missing Data Imputation

Missing data imputation for time series is a well-studied area, owing to the prevalence of time series applications and the frequent occurrence of missing values in recorded data, often stemming from issues with sensors [42]. It helps greatly improve the predictive capacities of Machine Learning models [43].

Traditional approaches to addressing missing data in time series datasets encompass several strategies. Data deletion, wherein incomplete observations are discarded is a popular strategy in many applications, however it may render the dataset incomplete and unsuitable for downstream applications, particularly when the missing data rate is high [49]. Statistical-based methods rely on statistical models to impute missing values, with simpler approaches leveraging basic statistics such as mean or median imputation [42], or employing likelihood-based computations [134]. Neighbor-based methods involve identifying the nearest neighbors of missing values based on other attributes, typically employing algorithms like KNN, and updating the missing values with the mean value of these neighbors [17], however this is only effective when neighbors are deemed reliable (anomaly-free data). Regression-based techniques entail learning a regression model from the nearest data points to predict missing values [135].

More recently, deep learning-based missing data imputation, particularly denoising autoencoders (DNNs), has emerged as a popular paradigm for denoising strategies [105, 14, 75, 137], vastly outperforming classic numerical approaches [75].

In this approach, incoming data is turned into a latent representation, through an encoder function, and subsequently reconstructed through a decoder function. Training data is deliberately corrupted by randomly setting intervals of sequential values to zero, mimicking the distribution of missing data. The training process is guided by the minimization of a loss function measuring the disparity between the original input and reconstructed output.

This process is further explained in section 3.2.2.

1.3.3 Time Series Forecasting

Early works on time series forecasting predominantly relied on statistical methods. Techniques such as Autoregressive Integrated Moving Average (ARIMA) and Exponential Smoothing (ETS) gained popularity due to their simplicity and effectiveness in capturing linear relationships and temporal dependencies within time series data [3]. These models established a robust foundation for time series analysis, particularly for short-term load forecasting tasks, and continue to be influential in certain applications. For instance, in [81], a seasonal ARIMA model outperformed several modern deep learning architectures in wind speed forecasting. However, these methods often encounter challenges regarding robustness, non-seasonal data, and prolonged forecast horizons [32].

As the demand for more robust forecasting methods grew, machine learning ML approaches began to emerge, significantly enhancing forecasting accuracy compared to traditional statistical methods. ML techniques excel in capturing non-linear dependencies and can effectively utilize larger datasets [23]. Prominent methods include random forests, boosting, and bagging ensembles of regression trees, as well as support vector machines for regression. These approaches also provide greater flexibility in feature engineering, allowing models to integrate various external factors beyond historical data, such as weather conditions, which contributes to improved load forecasting accuracy [5].

In recent years, the emergence of deep learning has revolutionized this field [76], with Recurrent Neural Network (RNN) based methods representing a significant breakthrough. While traditional RNN architectures face issues such as gradient vanishing or exploding over long sequences, advancements such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) have mitigated these limitations, as demonstrated in studies like [67] and [96] which focus on load forecasting for residential buildings. These models have proven effective in capturing long-term dependencies in time series data, demonstrating superior performance compared to tree-based algorithms [7, 146]. Additionally, attention-based mechanisms, which allow models to focus on relevant parts of the input data dynamically, have shown promising results in further enhancing forecasting accuracy [27]. Their self-attention mechanism is particularly efficiency in long sequence modeling, with noteworthy contributions including works like [142, 122]. However, concerns about the effectiveness of self-attention mechanisms have been raised, suggesting it may potentially cause temporal information loss [133]. Additionally, temporal convolutional networks, leveraging convolutional filters to capture local temporal features effectively, have emerged as another prominent approach such as in [100].

Recent work suggests that the most effective methods are hybrid approaches that combine multiple deep learning components to leverage their strengths into a single complex system. For this, we reference [44] that combines the LSTM with the CNN architecture to perform short term load forecasting and [142] which proposes an attention based encoder-decoder architecture that incorporates an auto correlation mechanism to effectively discover dependencies at the subseries level. Building upon this paradigm, we mainly note [79] stands out, demonstrating superior performance compared to multiple recent transformer-based methods. Further details about this model are provided in Section 3.2.3.

1.3.4 Automating Model Selection and Optimization

Hyperparameter optimization and Neural Architecture Search (NAS) have emerged as crucial components in the development and optimization of neural networks. Hyperparameter optimization involves systematically exploring hyperparameters, such as learning rates, batch sizes, and regularization parameters, to find the best configuration that maximizes the network's performance. Different techniques, including Grid Search, Random Search, Bayesian Optimization, and Genetic Algorithms, have been proposed to strike a balance between model complexity and generalization, improving the network's ability to learn and adapt [132].

In contrast, NAS focuses on automating the process of finding optimal neural network architectures. Traditional approaches relied heavily on domain expertise and manual trial-and-error, which proved to be time-consuming and suboptimal. NAS methods utilize search algorithms (e.g. Random Search [71]), Bayesian optimization [20], Reinforcement Learning [145], or Evolutionary Search strategies [114] to efficiently explore the vast space of possible architectures. These approaches automatically discover architectures that demonstrate state-of-the-art performance and generalization.

For instance, [91] makes use of an evolutionary computation based neural architecture search framework to find an optimal classification model for time series data. In [73], a Reinforcement Learning based NAS strategy was combined with Self-Imitation Learning [89] to perform out-of-sample anomaly detection in image datasets. Furthermore, it was used for pixel level anomaly detection in images in [64].

In a related vein, Automated Machine Learning (AutoML) encompasses a broader set of techniques that automate developing the various stages of a machine learning pipeline. This includes tasks such as data preprocessing, feature engineering, and model selection and optimization. An illustrative example of AutoML's applications can be observed in [74] where a search strategy based on the Sequential Model-Based Global Optimization algorithm [56] is employed to automatically generate an end-to-end outlier detection pipeline for time series data.

In addition, [141] adapts meta-learning for the model selection task to systematically find a suitable unsupervised model for outlier detection on a new unlabeled dataset based solely on historical models benchmarks and task similarity. Similarly, [140] uses meta-learning to automate the hyperparameter optimization process of unsupervised outlier detection models using the prior performance of hyperparameters on existing benchmarks datasets.

1.3.5 Federated Learning

A major challenge in the smart buildings field is the limited availability of large training datasets for deep learning models, particularly for new buildings or those with new occupants. Several researchers have attempted to address this by leveraging data from other buildings with similar load patterns. For instance, [78] enhances residential load forecasting accuracy for buildings with small datasets by introducing an attentive graph neural network that learns the relationships between electric loads across different houses, using a graph temporal convolution network for forecasting. Similarly, [47] proposes a sequence-to-sequence model within a transfer learning framework to tackle this issue. In another approach, [92] suggests a transfer learning method to improve load forecasts for a building with a limited dataset by using long-term measurements from other buildings with similar distributions. However, these works do not account for privacy concerns, as building occupants may not wish to share their data with others because energy usage data can reveal sensitive information about the occupants' habits and behaviors [104].

The idea to exploit multiple decentralized sources of private data to train a global model with no data exchange was first proposed in FedAvg [84]. This algorithm implements a server-clients architecture, where clients train local models on their own private data and only send the model updates back to a central server for aggregation. While FedAvg marked a breakthrough in privacy-preserving learning, it faces limitations in terms of computational costs, communication efficiency, and data heterogeneity. To address these challenges, several algorithms were introduced. FedDF [77] applies model distillation techniques to compress client models, reducing the data transmitted to the server and allowing clients to share knowledge without exchanging raw model parameters. FedProx [72] extends FedAvg by adding a proximal term to the local objective function, which constrains how much local models can deviate from the global model, mitigating issues with highly diverse data distributions and variable amounts of work being done by different devices due to differences in computational power and device availability. SCAFFOLD [60] introduces control variates to reduce update variance between clients, addressing the issue of client drift for non-IID clients, while FedNova [115] normalizes updates based on the number of local updates performed by each client, resolving imbalances caused by varying data quantities or computational power across clients.

Previous federated learning methods, such as FedAvg, focus on building a single global model for all clients, which often struggles to account for the diversity of data across clients. To address this, a new class of algorithms known as Personalized Federated Learning (PFL) has emerged, aiming to provide each client with a tailored model that better suits their unique data characteristics. These algorithms seek to leverage the shared knowledge across the federation while still maintaining personalization. In this context, Per-FedAvg [41], fine-tunes the global model using local data through meta-learning, allowing clients to rapidly adapt the global model to their specific data. PFedMe [108], introduces a regularization term based on the Moreau envelope that allows each client to learn a personalized model while keeping it close to the global model, thus balancing generalization and personalization. FedPer [9] tackles personalization by sharing only a subset of model layers among clients, while allowing others to remain client-specific, thus enabling personalization without sacrificing global knowledge. [51] proposes a personalized federated learning approach, where each client maintains both a private local model and a federated one that it shares across the network. It also proposes a loss function that mixes the weights of both models to induce a low-loss connection between theirs optima, allowing the client to sample from a weight subspace that minimizes the loss for both local and global objectives. [123] introduces adaptive parameter propagation, where clients propagate parameters based on a privacy aware distribution similarity estimation, and selective regularization, ensuring clients only incorporate other models' parameters if they positively impact their local model's performance. [136] optimizes the initialization of the local model before each FL training iteration by adaptively combining weights of the global and local models to better align with the local objective on each client. [52] introduces a privacy-preserving load forecasting framework for residential energy users by clustering users based on load pattern similarity without sharing raw data. The framework combines a federated user clustering method with hierarchical federated model training, enabling intra- and inter-cluster collaboration.

1.4 Contributions

This research has several contributions, listed as follows:

- **Neural Architecture Search for Anomaly Detection in Time Series Data of Smart Buildings: A Reinforcement Learning Approach for Optimal Autoencoder Design**: In this research, we propose to identify anomalies in time series data by training an autoencoder to learn to exclusively reconstruct patterns of normal behavior, and then flagging abnormal samples that it fails to reconstruct as they deviate from the learned behavior. We apply a Reinforcement Learning based Neural Architecture Search framework to automatically select an optimal autoencoder architecture for a given dataset and anomaly hypothesis, saving us the hassle of fine-tuning our model to every sensor type and building setting. More generally, we show that using this framework allows us to achieve efficient anomaly detection performance in time series data, despite using a simple deep learning model and without resorting to task-specific feature engineering.

This research was published in IEEE Internet of Things Journal [38].

- **Unsupervised Anomaly Detection and Imputation in Noisy Time Series Data for Enhancing Load Forecasting**: In this second work, we present an unsupervised memory-based anomaly detection framework for anomaly-contaminated time series data. This method extends existing research on anomaly detection, originally developed for image data [94, 59], to the context of time series analysis. Moreover, we develop a bi-LSTM-based recurrent denoising autoencoder neural network to replace detected anomalous patterns in the time series by plausible regular ones. We propose combining this model with the anomaly detection framework to create a data cleaning pipeline, improving the quality of noisy and anomaly-infected time series data for downstream tasks such as load forecasting. We then conduct a comprehensive study across different datasets of different scales and profiles to analyze how load anomalies impact the accuracy of different machine learning forecasting models and how their performance may be improved by using our anomaly detection and imputation pipeline to preprocess the data.

  This research was accepted for publication in Applied Intelligence [36].

- **Towards Efficient Federated Load Forecasting: Personalization Mechanisms and Their Impact**: In this third research, we propose a load forecasting approach for optimizing energy efficiency in smart buildings by leveraging a federated learning framework to exploit data from other smart buildings in a privacy-conscious way, thereby addressing the prevalent issue of data scarcity that prevents the training of accurate deep learning models while ensuring user privacy. Moreover, to address the non-IID nature of data, we explore and adapt recent advancements in personalization techniques for federated learning to the specific context of smart building load forecasting, comparing these approaches across multiple datasets to ensure personalized and accurate forecasting for diverse building profiles, ultimately recommending an optimal approach.

  This research was submitted to IEEE Internet of Things Journal [37].

1.5 Thesis Outline

The rest of this thesis is structured as follows.

- Chapter 2 presents the first contribution, detailing the reinforcement learning-based neural architecture search framework for anomaly detection in time series data. It describes the method and experiments conducted to demonstrate its effectiveness.

- Chapter 3 elaborates on the second contribution, introducing an unsupervised anomaly detection and imputation framework to preprocess noisy time series data. The chapter also presents experiments that assess how this system enhances the forecasting accuracy of machine learning models for buildings with anomaly-infected historical data.

- Chapter 4 presents the third contribution, describing a federated learning framework for load forecasting in smart buildings that aims to address data scarcity. It also explores personalization mechanisms to tackle the non-IID nature of building data, providing an evaluation of different personalization approaches.

- Chapter 5 summarizes the findings and contributions of the thesis. It discusses the implications of the research, its limitations, and suggests directions for future work in the area of anomaly detection and load forecasting in smart buildings.

Chapter 2

Neural Architecture Search for Anomaly Detection in Time Series Data of Smart Buildings: A Reinforcement Learning Approach for Optimal Autoencoder Design

2.1 Introduction

Integrating multiple smart devices within an Internet of Things network presents a multifaceted challenge, that includes considerations related to architecture, scalability, security, privacy and more [106]. This challenge is further complicated by the inherent susceptibility of IoT devices to issues like hardware malfunctions and transmission problems, resulting in data anomalies and introducing an additional layer of complexity for the effective use of the collected IoT data. This vulnerability, combined with imprecise measurements, leads to deficiencies in raw sensor data, including issues like noise, incompleteness, and faulty measures as discussed in [102]. Such erroneous data may lead to false interpretations, impacting the overall effectiveness of automated systems and impacting subsequent decision-making processes.

Therefore, the ML applications based on IoT sensor data are not robust to sensor anomalies as providing these ML models with such faulty data may deteriorate their capability [106], which in turn can lead to discomfort, energy inefficiency, and increased troubleshooting and operational costs.

For this end, we propose that identifying such irregularities in building meters will enhance the robustness of ML based applications and data driven decision-making and will allow early detection of failing sensors and systems.

Moreover, for power monitoring sensors, detecting anomalies in near real-time data allows energy suppliers to promptly identify technical issues in the distribution network, eliminating the otherwise reliance on customers for reporting outrages [30], hence enabling timely interventions and preventing minor issues from escalating into significant problems. This also enables the detection of non-technical abnormal patterns linked mainly to energy theft [131].

Therefore, we propose an anomaly detection framework for IoT sensors which exploits the predominant regular behavior that is easier to collect in a smart building context where anomalies are rare and hence difficult to label for training a supervised ML approach. This method works by training an Autoencoder (AE) neural network on reconstructing anomaly-free samples, guided by the minimization of the error between the original and the reconstructed sequence. Then, as data sequences containing anomalies deviate from the learned regular behavior, the reconstruction error will be higher than the ones observed during training. Consequently, setting a threshold on the reconstruction error allows us to detect abnormal sequences for which this error is excessively high.

Nevertheless, the fidelity of the reconstruction depend on several hyperparameters of the neural network architecture such as the number of layers, their type and their parameters, and misconfiguring them may lead to a general low reconstruction accuracy or in other cases, to accurately replicating both normal and abnormal patterns hindering their error-based separability.

This parameter optimization process requires extensive domain knowledge and heavily depends on the building's configuration and the sensor type. To overcome this difficulty, we treat this task as a sequential decision-making process, and apply a Reinforcement Learning (RL) based Neural Architecture Search (RLNAS) approach to automatically find the optimal autoencoder architecture without any prior knowledge about the building's parameters or any assumption about the anomalies.

Moreover, [63] highlighted that occupant monitoring accounts for 24% of a building's energy savings, with weather forecasts contributing 7%, as they allow optimizing the control of HVAC systems. Therefore, we evaluate our anomaly detection framework on sensor data relating to indoor occupancy estimation and to records of weather meters. Furthermore, we showcase how filtering anomalies from the training data of an occupancy estimation ML model can enhance its performance.

The rest of the chapter is organized as follows: in Section III, we present how our proposed system works. We further detail in Section IV the methodology we adopted, including a description of the used datasets and the setup for the experiments we conduct. We also analyze the obtained results and evaluate the applicability of our system in enhancing the performance of occupancy estimation ML models. In Section V, we finally discuss the achieved performance and go through the limitations of our method.

2.2 System Overview

In this section, we propose an automated time series anomaly detection framework to find the optimal autoencoder model for a given dataset. We first explain how autoencoders can be used to detect anomalies through reconstruction error. Then we detail the two essential components of our neural architecture search framework: the search space, i.e. the architecture space to be explored, and the search strategy, i.e. the Reinforcement Learning algorithm used to guide the search.

2.2.1 Reconstruction Based Anomaly Detection

In this study, the selection of an Autoencoder neural network as our anomaly detection model was motivated by the fact that, as an unsupervised learning technique, it doesn't require labelled anomalous data (except for evaluation purposes). This characteristic makes them particularly well-suited for scenarios where labelling anomalies is costly or impractical, such as our case. Additionally, neural networks have the remarkable ability to autonomously learn features from raw input

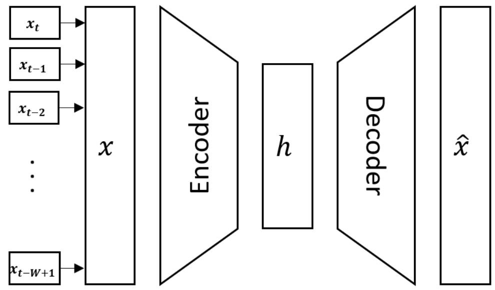


Figure 2.1: Autoencoder Architecture

data, eliminating the need for laborious and time-consuming manual feature engineering conducted by domain experts.

The Autoencoder is a neural network that employ an encoder enc to compress a given 1-D sequence of size w, $x = (x_t, x_{t-1}, ..., x_{t-w+1})$ into a lower-dimensional representation h in the latent space through consecutive convolution and pooling operations. A decoder dec is then employed to reconstruct the original input sequence from its latent representation by upsampling it back to its original size through transpose convolution. Incorporating nonlinear transformations and a bottleneck layer enables autoencoders to capture essential patterns while discarding irrelevant information, allowing them to reconstruct the input sequence x while effectively ignoring the noise that may affect it. Its general schematic is presented in Fig. 2.1.

This learning process is driven by minimizing a reconstruction loss error, which measures the discrepancy between the input sequence and its reconstructed output. In our approach, we specifically utilize the Mean Squared Error loss, as it emphasizes the impact of outlier data points, making it well-suited for anomaly detection.

$$L(x, \hat{x}) = \sum_{i=1}^{W} (x_i - dec(enc(x_i)))^2 \tag{1}$$

During its training, the autoencoder is exposed to regular, non-anomalous time series patterns. It learns to reconstruct them accurately with minimal reconstruction error. consequently, it will not be able to faithfully reconstruct the abnormal patterns that it may encounter when deployed in a real-world noisy environment as they deviate from the regular patterns it learned, therefore reconstruction error will be noticeably higher and by setting a threshold on the error, anomalous time series can be identified.

2.2.2 Search Space Design

In our proposed approach, we employ a RL agent to make a series of decisions for building the Autoencoder neural network, one layer at a time, including determining whether a particular layer should be incorporated into the architecture and, if so, specifying its parameters.

However, arbitrarily choosing neural layers may lead to asymmetric architecture that produce a sequence $dec(enc(x))$ with a different dimensionality than x, preventing us from calculating a meaningful error function.

To avoid this, the kernel size of the transpose convolution operation allowing to upsample the sequence length back to its original size needs to be set according to the stride, padding and kernel size of the previous convolution and pooling operations. Therefore, when designing the search space for our RLNAS framework, we only include the parameters to be explored for the encoding network and then construct the decoding network by stacking the adequate layers to ensure the symmetry of the whole architecture.

Therefore, the autoencoder architecture A is divided into N encoder building blocks encapsulating a set of fundamental repeatable operations, followed by N corresponding decoder building blocks (Eq. 2) where each encoding block is composed of sequential layers: convolution, pooling, activation and normalization (Eq. 3). To each encoding block enc^i corresponds a decoding block dec^i composed of sequential transpose convolution, normalization and activation (Eq. 4) that is responsible for reversing the dimensionality reduction that occurred in the encoding process at the pooling layer i. This is done by selecting the adequate size of the Transpose Convolution's kernel while the choice of number of convolution channels, normalization type and activation function are the same as the encoder's as they do not affect the dimensionality of the produced representation.

$$A = \{enc^1, ..., enc^N, dec^1, ..., dec^N\} \tag{2}$$

$$enc^i(x; w_i) = Act(Norm(Pool(Conv(x))) \tag{3}$$

$$dec^i(x; w_i) = Act(Norm(TConv(x)) \tag{4}$$

where x denotes the input sequence, and w denotes the trainable weight matrix.

Essentially, we select the following set of potential parameters to be explored by the RL agent.

- $Conv()$: in each layer, the number of channels produced by the convolution operation is 1, 2, 4, 8, 16, 32 and the size of the convolution kernel 3, 5, 7, 10, 15.

- $Pool()$: the type of pooling in each layer, either none, max pooling or average pooling, as well as the pooling kernel size 3, 5 or 7.

- $Norm()$: the normalization layer type, including batch normalization, instance normalization, and no normalization.

- $Act()$: the non-linearity is chosen to be either sigmoid, tanh, relu, elu or none.

The choice of parameters to include in the search space is made arbitrary, and we tried to include as many possible parameters as possible since we are simulating the usual Reinforcement Learning framework where the agent has no prior knowledge about which decisions may be more favorable for the specific situation. And exposing it to a wide range of parameter choices allows it to learn through trial and error, gradually discovering potentially beneficial configurations that may not be immediately apparent or intuitive.

However, the search space size is limited by computational efficiency, and we don't include large values for the kernel size of the convolution and pooling operation as this may in some cases produce latent representation in the bottleneck layer that are too small to ensure the feasibility of the reconstruction according to the set stride value and maximum number of building blocks.

Beside the neural network architecture, other parameters like the optimization algorithm or the number of epochs are manually fixed, meaning that we are looking for an optimal architecture under a predefined set of constraints.

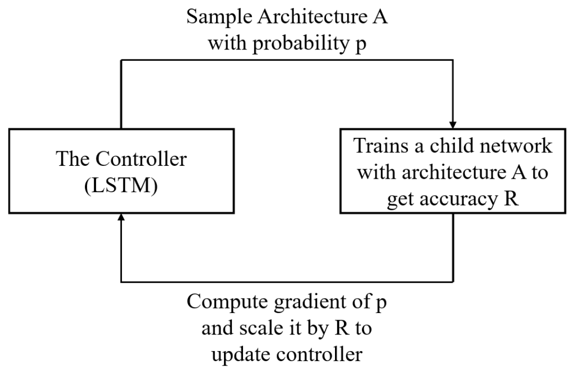


Figure 2.2: An Overview of Neural Architecture Search [145]

2.2.3 Neural Architecture Search Strategy

Neural Architecture Search (NAS) is a prominent approach to automating the search for an optimal neural network architecture. It can be viewed as a meta-learning process, where a controller (RL agent) is instructed to explore a given parameter search space by iteratively sampling and training child neural networks and observing their performance (Fig. 2.2).

From a theoretical perspective, NAS can be viewed as a Markov Decision Process (MDP) because of its sequential decision-making process under uncertainty.

In this context, the history H of the tested neural architectures represents the states within the MDP framework. Then at each step l, a reinforcement learning agent, defined as an LSTM neural network, calculates decision probabilities and accordingly samples an action a_l that entails choosing hyperparameters and adding one layer at a time to the child autoencoder model according to its current policy $\pi(a_{1:l-1})$. Thus, we gradually build the model architecture allowing a fine-grained search, where each decision (e.g., selecting a layer type or its hyperparameters) is made sequentially based on the previous decisions for the current model and on the insight it gets from the selection of the best models discovered so far.

The LSTM-based agent receives feedback on the performance of the current architecture corresponding to $(a_1, ..., a_l)$ in the form of a reward signal R_l, which serves as a guiding factor for decision-making. We chose this reward to be an evaluation of the child model's performance on the validation dataset D_{val} after its training.

Because the chosen reward signal R_l is not differentiable, we have to use a policy gradient method to update the agent's neural network weights. So in alignment with established practices in the field, and specifically inspired by [145], we opt to employ the well-known Reinforce algorithm [118] in our RL framework to update the weights of the controller neural network according to the following rule:

$$\theta \leftarrow \theta + \alpha \nabla_{\Theta} \log \pi(a_{1:l-1}) G_l \tag{5}$$

where G_l is the discounted reward, used to take into account the fitness achieved by the subsequent decisions within the same architecture for cases where the choice of the last layers significantly improve the fitness of the whole architecture.

As a result, for the next episodes, the RL agent will give higher probabilities to decisions that receive high anomaly detection scores. And the best architectures discovered so far will be passed on to the RL agent as input in the next episodes.

This way, through iterative exploration and reward-based learning, the agent strives to identify the most suitable autoencoder configurations for the given dataset and learning parameters by maximizing its expected reward.

We further outline in Algorithm 1 the steps for the RL-based neural architecture search.

Algorithm 1 Autoencoder Neural Architecture Search

Input: Datasets D_{train}, D_{val}, search space S
Output: Optimal autoencoder architecture A

```
Initialize controller weights θ, search history H_0
for each episode e do
    for each layer l do
        Sample an action a_l ~ π(a_{1:l−1}; θ) from S given H_e
        Train the architecture A(a_1, ..., a_l; ω) on D_train
        Infer A on D_train to select the anomaly threshold T
        Evaluate A on D_val to obtain the reward R_l
    end for
    for each decision l do
        Calculate the discounted reward G_l ← Σ_{k=l+1}^{L} γ^{k−l−1} R_k
        Update the controller weights θ ← θ + α∇_Θ log π(a_{1:l−1}; θ)G_l
    end for
    Add A to H_e
end for
```

2.3 Experiments

In this section, we start by describing the custom dataset that we created to evaluate our approach and how it was created. Then we detail our experimental setup and the results we obtained. We also provide a case study for the best autoencoder architecture discovered by the RL agent to gain insight into the decisions it learned to perform through its training. Finally, we illustrate how anomalies present in IoT sensors measurements can adversely affect the performance of machine learning models used in smart buildings and hence showcase a potential application of our proposed system.

2.3.1 Used Datasets

As of our current knowledge, while several datasets of regular time series sensor behavior in the context of smart buildings exist, no labelled anomaly detection dataset is available. Consequently, we resort to creating our own data from an existing one by generating artificial anomalies from regular data instances. To achieve this, we select the following datasets of time series sensor data and adapt them for the anomaly detection task by injecting synthetic anomalies into the originally anomaly-free measurements.

Office Room Dataset We consider the dataset provided in [8] as our primary data source. The dataset was originally curated for the purpose of estimating occupancy in intelligent buildings. It contains time series data recorded within a professor's office space (office H358) at the Grenoble Institute of Technology, which may accommodate at most a total of four individuals. The data were captured using multiple common non-intrusive sensors with a variable sampling rate (seconds to few minutes), spanning a total duration of almost three years between 2015 and 2018. It encompasses various informative features, including luminance, temperature, humidity, power consumption and co_2 concentration, among others. Additional application of such sensors include the control of HVAC systems, load monitoring and energy optimization.

Weather Stations Dataset We also gather a diverse set of weather data from five different weather stations located within the city of Montreal (station IDs shown in Table 2.2). By scraping information from the Weather Underground website [1], we were able to capture the available historical records of temperature, pressure and humidity for each station, spanning the years 2020 to 2023. The data sample rate for our study was set at 5 minutes, providing a detailed and granular view of the weather conditions throughout the specified period.

In addition to the previously mentioned applications of anomaly detection in sensor data, another particularity of these considered datasets is the critical role they play in energy-related applications as various systems rely on such data to perform tasks like load forecasting, energy disaggregation, and HVAC system optimization.

Anomalies in the measured data can adversely affect the performance of machine learning and deep learning models utilized in these systems. Such anomalies can lead to increased errors in predictions and subsequently hinder the overall performance of the systems. Hence, detecting and effectively handling anomalies in these datasets is crucial to maintain the reliability and accuracy of the models, thereby ensuring the effectiveness of energy-related applications and minimizing potential disruptions in building performance and user comfort.

2.3.2 Anomaly Generation

For generating realistic anomalies given regular data patterns, we adopt the approach described in [110] which allows generating two different types of anomalies. First, local outliers, which corresponds to single instances outlying with respect to their local neighborhoods and represent contextual anomalies which data points that, when observed individually, appear to be within the range of expected values for the signal. However, when examined in the context of neighboring observations, the overall sequence is deemed abnormal. The second type is anomalies that are synthesized by inserting global outliers (outliers scattered across the whole regular instance space) into regular data. They correspond to point anomalies that may be produced by faulty sensing equipment or short period events [30].

To generate the former kind, a Gaussian Mixture Model (GMM) is first fit to the regular data

[1] https://www.wunderground.com/

distribution, then a new GMM with the same parameters but with a scaled covariance matrix is built and used to generate outliers, resulting in data points that deviate from the regular distribution. As for the latter, they are sampled from a multivariate uniform distribution whose planes are bounded by values 5% higher than the maximum and lower than the minimum of each attribute of the regular dataset. Finally, a K-Nearest Neighbors classifier is used to detect and filter regular artificial data points from outlier artificial samples representing anomalies because their distributions overlap.

Generating these two types of outlier points allows us to synthesize all conceivable types of single-point anomalies. Indeed, [30] indicates that for time series data of IoT sensors, collective and point anomalies represent the two possibilities for single-point anomalies, and that the rest of anomalies consist in pattern anomalies corresponding to regular data points that only may seem as anomalies if considered as a group. The scope of this research does not cover this last kind.

This approach is used to generate anomalous data points that we randomly insert into the regular time windows, thus replicating real-world scenarios where anomalies can arise sporadically. We insert one anomaly at most within a single time window of 100 data points, with a time-window contamination rate of 20%.

In figure 2.3, we show two examples of such artificial anomalous samples. The top plot represents a contextual anomaly within pressure measurements, whereas the bottom plot illustrates the contamination of temperature data with global outliers.

2.3.3 Data Pre-processing

As we aim to find whether the neural architecture search framework is able to find a suitable autoencoder given any set of constraints on the anomaly definition and training parameters, we incorporate minimal pre-processing on the data and exclude employing elaborate task-specific feature engineering.

In fact, since the considered dataset is clean, we don't have to leverage any pre-processing method apart from normalization. Normalization is an essential step in our case because without it, the reconstruction of the model will be largely affected by the additive bias it learns, which models the mean value of the time series window at the time of measurement. It will therefore fail to generalize its reconstruction to time series windows taken from days or periods of the day with

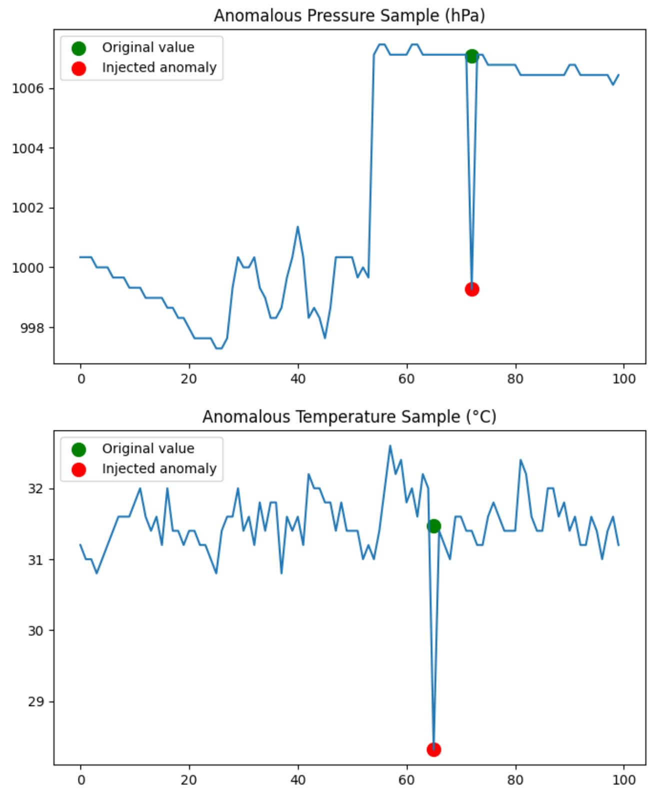


Figure 2.3: Example of a contextual anomaly synthesized by injecting a local outlier (red point) into regular pressure measurements (top) and point anomaly synthesized by injection a global outlier into a temperature sensor's measurements (bottom).

different mean values for the considered sensor data, for instance.

2.3.4 Methodology and Experimental Setup

To process the data, we divide the time series data into time windows of size 100, meaning that we observe about 8h of the sensor data at once and try to find whether an anomaly occurred within this time frame. The time windows are captures using a sliding window with a stride of 50. But for the first dataset, different sensors have different sampling rates and therefore their time series data have varying lengths. We end up for example with a total count of 25000 samples for the temperature and humidity sensors but only 1400 time window sample for the illuminance sensor. As for the second dataset, the weather stations have different a different number of available historical records, hence the number of sample we obtain range from about 3000 to 8000 sample.

In all cases, we reserve 20% of available data for validation (evaluation of discovered architectures' fitness during the RLNAS process) and another 20% for the final test of the selected top architecture on unseen observations.

Moreover, to build the RL agent, we employ a hybrid neural network architecture that is composed of an embedding layer of 128 nodes, a single LSTM layer of the same size and multiple fully-connected layers for independently taking each decision. Its weights are randomly initiated, and it is trained using the ADAM optimizer, with a learning rate of 10^{-3}. This search process is conducted for a total of 200 episodes, and the search history H maintains a record of only the top 20 best-performing models [2].

Each autoencoder model architecture is represented by a sequence of tokens corresponding to the parameters choices. We conduct neural search for architectures of 1, 2 and 3 encoding blocks, with a total of 6 hyperparameters per block. These child models are trained for 10 epochs using a learning rate of 10^{-3}, a batch size of 32 and a momentum of 0.9 using the Stochastic Gradient Descent (SGD) algorithm. The convolution kernel's stride and padding are fixed respectfully to 1 and 2 in order to reduce the search process's complexity by reducing the number of parameters per single layer to 2 (number of channels and kernel size).

In our proposed methodology, we make the explicit decision to set the anomaly threshold to the 99^{th} percentile of the reconstruction losses observed during training. Setting the threshold at such a high percentile allows us to prioritize precision in anomaly detection, thus minimizing the occurrence of false positives (i.e. instances where regular patterns are incorrectly classified as anomalies) to be at least less or equal to 1% of train data. This is done because false positives are generally the most undesirable outcome in anomaly detection tasks as they lead to unnecessary investigations or actions, which can be costly in terms of time, resources, and manpower.

This threshold is calculated by inferring the model again on the train data after it has been trained. We then evaluate it on the validation data by computing the reconstruction loss for the new sensor observations and comparing them with the threshold.

[2] Code for this implementation has been published and is available at the following link: github.com/MaherDissem/RLNAS-for-Autoencoder-based-Anomaly-Detection-in-Time-Series-Data

Given that this is an anomaly detection task, the data inherently exhibits significant class imbalance because of the rarity of anomalous events. Therefore, we utilize the $F1$ score (Eq. 6) to measure the autoencoder's fitness. The $F1$ score is a balanced metric that combines precision, which measures the accuracy of anomaly predictions, and recall, which assesses how well the model detects actual anomalies. It helps strike a balance between minimizing false alarms and ensuring that true anomalies are effectively identified.

$$F1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

The evaluation metric is then given to the RL agent as a reward signal to update its weights and guide the search.

2.3.5 RLNAS Comparison Baseline

We can conduct a comprehensive assessment of our combined framework's anomaly detection performance using metrics such as the $F1$ score, precision, and recall. However, we also ought to separately evaluate the RLNAS framework's effectiveness in discovering adequate neural network architectures.

But, given our decision not to make assumptions about the optimal hyperparameters for specific building configurations, we must work within a large parameter search space where testing all conceivable combinations to pinpoint the optimal performance would be excessively demanding in terms of computational resources (the search space mentioned in Section 2.2.2 would encompass approximately 4000 potential combinations for a single encoder block).

Consequently, we opt to employ a Random Sampling-based approach for Neural Architecture Search (RSNAS). This approach allows us to assess whether incorporating RL-driven decision-making in the search process actually helps uncover neural configurations that outperform random guessing.

This second approach consists of randomly selecting neural parameters from the same search space and recording the evaluation $F1$ score of each sampled architecture on the validation data.

Table 2.1: Performance comparison of using RLNAS versus using RSNAS to find a suitable Autoencoder architecture for anomaly detection on the H358 office dataset

Sensor type	RLNAS			RSNAS		
	F1 score	Precision	Recall	F1 score	Precision	Recall
Temperature	100	100	100	99.7	99.4	100
Humidity	99.7	99.5	100	93.3	99.3	88.1
Power usage	93.0	86.9	100	39.5	82.7	25.9
CO_2 concentration	57.2	96.2	40.7	59.6	96.6	43.1
Illuminance	05.8	50.0	03.2	03.5	12.5	02.0

We repeat this process for the same number of iterations as for RLNAS. Then, we select the best-discovered architecture and evaluate its performance on the test data.

2.3.6 Results

Table 2.1 showcases the achieved performance in terms of precision, recall and $F1$ score of the best architecture found by the RLNAS framework for anomaly detection in the office sensors dataset.

Overall, we obtain higher precision scores than recall scores on the test data because our approach privileges precision (low false positives) since we set the anomaly detection threshold as the 99^{th} percentile of the reconstruction losses observed during training. However, we are more interested in the $F1$ score, which is more suitable for describing the performance of classification tasks in imbalanced domains like our case.

We achieved excellent $F1$ score detection performance for the temperature, humidity and power consumption sensors. These sensors exhibit relatively simple patterns with low variation around a mean value, making the detection of anomalies relatively straightforward (see Fig. 2.3). Consequently, we obtained the best performance for these sensors.

Furthermore, we obtained satisfactory results for the co_2 concentration sensor, although the recall rate in classification is slightly lower. However, upon analyzing the data, we observe repeating day and week level patterns in this sensors' data, indicating the potential for improved performance through a more tailored approach. This way, the performance of our method can be further enhanced in the future by placing emphasis on capturing the seasonality expressed by such sensors.

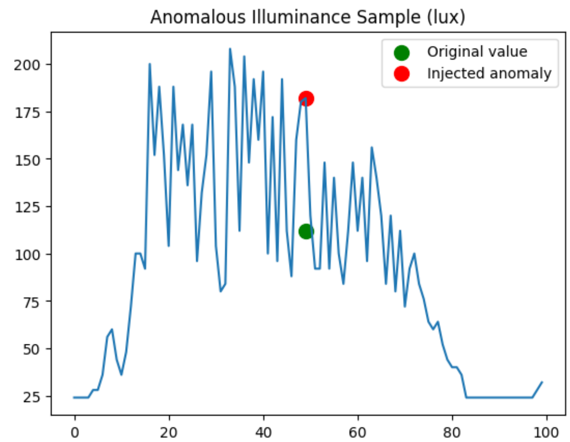


Figure 2.4: Example of an anomalous sample generated for the illuminance sensor's data by replacing the original value (green) by a local outlier (red).

On the other hand, our system yielded poor results when tasked to detect anomalies for the office's illuminance sensor. This can be attributed to the significant fluctuations, rapid up and down variations, and spikes present in this particular time series as illustrated in Fig. 2.4. These characteristics pose challenges for both human observers and AI models in distinguishing between artificially generated data points and those that correspond to regular time series behavior. In fact, because of these fluctuations, the autoencoder may learn to only reconstruct the overall trend of the sequence, resulting in high reconstruction errors for regular data that are barely affected by the presence of artificial outliers. Or it may learn to accurately reconstruct the whole sequence, but that leads the autoencoder to learn to reconstruct any complex signal pattern, including the sequences containing outliers, as they are hardly discernible from regular data points. However, this problem is specific to this particular type of data and anomaly definition. And to address this particular situation we might require more task-specific feature engineering to obtain satisfactory performance using simple deep learning models such as the autoencoder, which is out of the scope of our current study.

Furthermore, we present the performance evaluation of the proposed model architecture for each station in the second dataset, which comprises data captured from weather stations.

Table 2.2 provides a comprehensive overview of the achieved results, with $F1$ scores ranging from 89.0% to 100% for the temperature measurements, 61.1% to 92.8% for the humidity sensor's data and 83.6% to 97.2% for the pressure sensor's data. Moreover, the remarkable precision scores

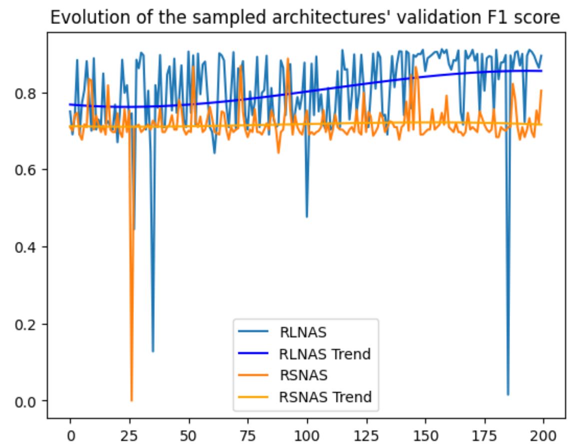


Figure 2.5: Evolution of the sampled architecture' $F1$ score on the validation data: RLNAS versus RSNAS

achieved demonstrate the model's ability to minimize false positive detections, while the high recall scores indicate its capacity to capture a significant portion of the actual anomalies.

Additionally, when compared to the Random Search algorithm, RLNAS achieves higher $F1$ scores overall for both datasets, outperforming the baseline by a significant margin. The main difference between the two algorithms is that while for RSNAS the whole architecture is randomly sampled at once, RLNAS samples the architecture one parameter at a time which provides an evaluation of the effect of each parameter individually in comparison to previous decisions, allowing to find better layer combinations. Moreover, the RLNAS' sampling distribution is affected by the outcome of previous episodes, whereas RSNAS always samples from a uniform distribution.

As seen in Fig. 2.5 which illustrates the evolution of the agent's reward (sampled architecture's $F1$ score on the validation data), comparing it to the RSNAS baseline for the IMONTR172 humidity sensor, the architectures sampled by RLNAS are overall more varied and achieve a larger range of scores. Furthermore, while RSNAS architectures values exhibit relatively consistent scores around a central point, we observe a noticeable positive trend in the scores of the RLNAS-sampled architectures.

This validates the assumption that that incorporating experience from past trials into the future decision-making process in the form of a RL agent's reward lead to gradually discovering better neural configurations outperforming repeated random guessing even for a significant number of iterations.

However, this performance gain comes at a price, as the RLNAS search process takes generally longer to complete. This is a consequence of the need to gradually build an architecture by evaluating the performance impact of choosing one parameter at a time. The training time is heavily influenced by parameters like the available computational power, the dataset size and the maximum size of the AE architecture.

The average runtime of the RLNAS search for the two datasets is depicted in Fig. 2.6, where it is compared to RSNAS for various target architecture sizes during a search process of 200 iterations conducted on an Nvidia RTX A6000 GPU. Notably, the RLNAS process exhibits a significantly longer completion time compared to RSNAS. This difference becomes more pronounced as the target architecture size increases, primarily because a larger search space has a more substantial impact on the run time of the RLNAS process due to each decision being separately evaluated instead of evaluating the whole architecture at once like for the Random Sampling method. Moreover, the RLNAS process takes significantly more time for the H358 dataset, which comprises approximately 9.5 times more data points in comparison with the weather dataset due to its smaller sampling rate.

Overall, the extended runtime proves to be worthwhile as the obtained results demonstrate the superior capabilities of the discovered model architectures in accurately detecting anomalies within IoT sensors' data, indicating the ability of the RLNAS approach to discover neural configurations optimized to capture the specific anomalies contained within each type of sensor data.

2.3.7 Architecture Optimality Analysis

To comprehensively assess the sensitivity of the RLNAS sampled architecture, we propose to investigate the impact of varying neural parameters on the anomaly detection performance to see if for this particular anomaly definition, some hyperparameters inherently perform better than others. Specifically, we focus on the IMONTR17 weather station's pressure sensor and to minimize the interdependence of multiple architectural decisions, we deliberately choose a small architecture of one block. We employ the RLNAS framework to identify an optimal neural architecture, then individually vary parameter decision to see if the architecture could be further improved. The outcomes of this experiment are visualized in Fig. 2.7, illustrating the effects of different parameter values on

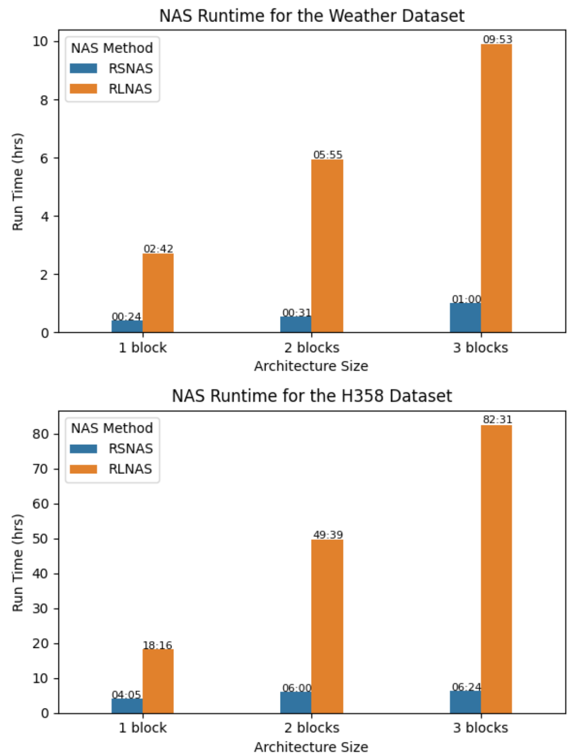


Figure 2.6: Runtime comparison between RLNAS and RSNAS for the two datasets for various model sizes

overall anomaly detection performance for the particular sensor.

We observe in this case that the discovered architecture is not significantly sensitive to parameter variation, as the $F1$ difference between the optimal decision and the RLNAS choice varies from 0 to 3.1%. Furthermore, attempting to enhance performance by constructing an autoencoder replacing all suboptimal parameters by the optimal ones at once results in lower performance. Specifically, this autoencoder achieves an F1 score of only 0.71, which is notably lower than the initial F1 score of 0.81.

Overall, this highlights the inefficiency of deriving conclusions from the evaluation of individual hyperparameters, as this approach neglects the intricate interdependencies that exist among multiple decisions within the same architecture, especially for large architectures. Moreover, the practice of selecting the optimal layer and sequentially seeking the best consecutive parameters to build a full AE model proves inefficient. This methodology essentially mimics a zero-exploration reinforcement learning strategy, which, by its nature, is suboptimal due to its failure to explore a

Table 2.2: Performance comparison of using RLNAS versus using RSNAS to find a suitable Autoencoder architecture for anomaly detection on the weather stations' dataset

Sensor type	Station ID	RLNAS			RSNAS		
		F1 score	Precision	Recall	F1 score	Precision	Recall
Temperature	IWESTM40	100	100	100	91.1	100	83.5
	IHAMPS1	96.8	100	93.8	90.8	100	83.1
	IMONTR17	89.0	85.6	92.8	75.9	64.0	93.5
	IMONTR172	95.9	96.3	95.5	91.4	92.6	90.1
	IQUEBECM21	95.9	92.2	100	80.3	96.3	95.3
Humidity	IWESTM40	86.7	100	76.6	66.2	100	49.3
	IHAMPS1	93.7	100	88.2	57.6	100	40.4
	IMONTR17	61.1	59.4	62.2	60.4	81.2	48.1
	IMONTR172	90.0	100	81.8	74.9	100	59.4
	IQUEBECM21	92.8	100	86.6	70.2	100	54.1
Pressure	IWESTM40	97.2	96.6	97.7	96.0	96.5	95.5
	IHAMPS1	93.0	94.8	91.1	87.9	100	78.4
	IMONTR17	83.6	76.0	92.8	80.3	69.6	95.0
	IMONTR172	96.1	95.2	97.0	87.6	89.0	86.4
	IQUEBECM21	95.8	100	92.0	93.6	100	88.0

diverse array of possibilities. Essentially, RLNAS provides a solution to overcome this limitation by adopting a holistic and exploratory optimization process that considers the interconnected nature of architectural decisions, wherein the RL agent's hyperparameter sampling policy integrates insights from previous decisions within the same architecture.

Besides, while this example presents a compelling case, we extend our investigation by replicating the same experimentation across multiple sensor time series to rigorously assess the generalizability of these findings. Our comprehensive analysis reveals that further optimizing a single parameter at a time for a two-block architecture results in a performance enhancement of 3.5% on average across all the weather dataset time series. This illustrates that the RLNAS strategy is arguably efficient at finding optimal configurations, leaving little room for manual improvement unless we consider a radically different solution.

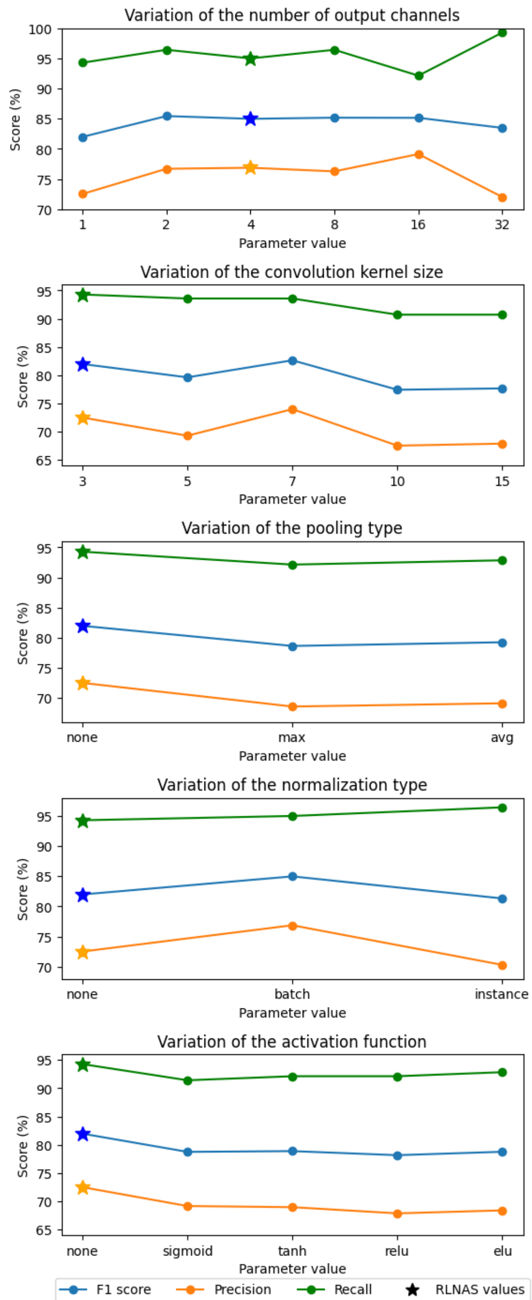


Figure 2.7: Effect of varying a single hyperparameter on the anomaly detection metrics of the IMONTR17 pressure sensor

Table 2.3: Parameters of the encoder of the best discovered architecture for anomaly detection in the data of the office humidity sensor

Block	Parameter	Decision
Block 1	Convolution output channels	4
	Convolution kernel size	10
	Pooling type	None
	Pooling kernel size	-
	Normalization type	Instance
	Activation type	None
Block 2	Convolution output channels	8
	Convolution kernel size	3
	Pooling type	Average
	Pooling kernel size	7
	Normalization type	Batch
	Activation type	Sigmoid
Block 3	Convolution output channels	16
	Convolution kernel size	3
	Pooling type	None
	Pooling kernel size	-
	Normalization type	Batch
	Activation type	Tanh

2.3.8 Case Study

During the training phase, the agent systematically explored diverse architecture configurations for the anomaly detection autoencoders. As training progressed, it gradually discovered more effective configurations, as evidenced by higher $F1$ scores, thereby improving the performance of the anomaly detection system. In this section, we further explore the architectures it discovered and how the learned decisions affect the overall performance of the child model.

In particular, we first provide insight into the anomaly detection architecture proposed for the humidity sensor located within the office space and whose measurements are centered around $36g.kg^{-1}$, with a standard deviation of $5.5g.kg^{-1}$.

The validation fitness ($F1$ score) of the different configurations sampled by the agent during its training range from 65% to 94%. Table 2.3 display the discovered network configuration which achieving the highest $F1$ score.

We build this proposed model and run it through the proposed anomaly detection pipeline under the methodology that we previously detailed in order to understand how it reconstructs both normal and anomalous samples, and thus gain insight into the rules learned by the RL agent during its training.

In figure 2.8, we show and compare how the model behaves when tasked to reconstruct a sample of normal sensor behavior and an artificial anomalous data sample that we generated. To our surprise, we notice that the autoencoder model could achieve good performance without fitting to the normal sequence behavior. Indeed, despite not faithfully replicating the patterns of the normal samples, the MSE reconstruction loss remained lower than the selected error threshold in this example.

In fact, through testing other suboptimal architectures, we found that models that successfully reconstructed input sequences with high fidelity exhibited lower $F1$ scores. This is because, these particular architectures learn to perfectly reconstruct complex patterns and are therefore also able to closely reconstruct anomalous sequences, resulting in a higher false negative rate (anomalies detected as normal samples). This might go against the intuitive idea one may have of generating an autoencoder that's perfectly able to replicate normal behavior to ensure that the reconstruction of normal sequences yield a low reconstruction error.

Second, we study the top architecture discovered for the temperature sensor of the IWESTM40 weather station, which is located in Westmount, a municipality on the Island of Montreal. The regular values of this sensor range between -17.8 and 34.2 Celsius degrees. Fig. 2.9 illustrates the reconstruction of regular and anomalous normalized data instances by the autoencoder neural architecture found by our RLNAS framework (displayed in Table 2.4) which results in a $F1$ score of 100%.

In contrast to the previous example, the time series data of this second use case are vastly different, as the temporal evolution of temperature is overall smoother. Nevertheless, the top proposed model doesn't reconstruct such smooth patterns, instead the reconstructed sequences exhibit irregular curves characterized by significant fluctuations which only vaguely follows the trend of the regular sequence.

In this case, the reconstruction process fails to capture individual outlier data points. Despite this, these outliers have a drastic effect on the reconstructed series, as they significantly amplify its fluctuations. This may be due to outlier points influencing both the maximum and average pooling

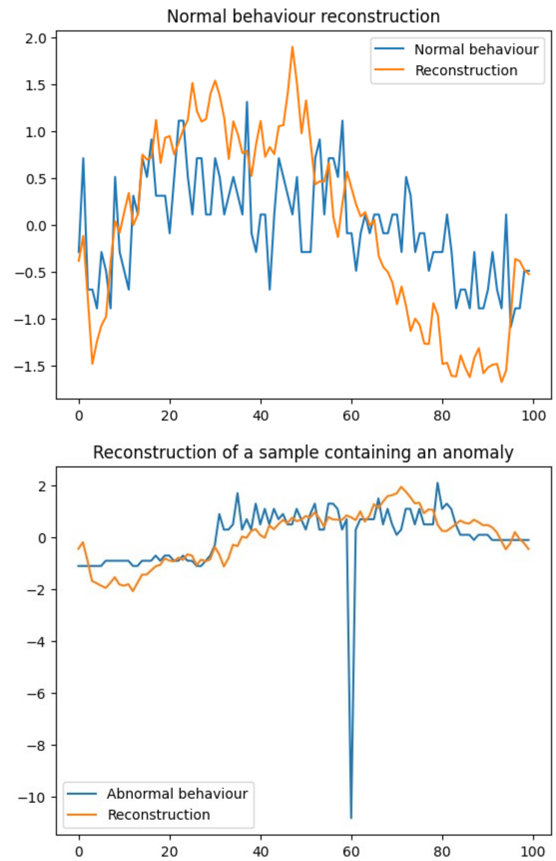


Figure 2.8: Reconstruction of a normal (top) and anomalous (bottom) data sample by the top discovered autoencoder architecture for the data of the office's humidity sensor

calculations of adjacent points, which results in a higher reconstruction loss compared to regular samples, making this architecture particularly well-suited for anomaly detection.

Furthermore, this architecture differs significantly from the previous one as it is smaller yet incorporates a greater number of convolution kernels (higher channel count) with different kernel sizes and uses different pooling and activation functions.

In fact, on this data and under the same training conditions, the previous architecture (Table 2.3) yields a much lower $F1$ score of 49.28% due to high false negatives, as it predicts most data samples as regular, anomaly-free data (100% precision and 32.7% recall). This illustrates that a same neural network may perform very well for a particular type of data while performing poorly for a different kind of patterns. This highlights the difficulty that a developer may face to identify a suitable neural

Table 2.4: Parameters of the encoder of the best discovered architecture for anomaly detection in the temperature measurements of the IWESTM40 weather station

Block	Parameter	Decision
Block 1	Convolution output channels	32
	Convolution kernel size	3
	Pooling type	Max
	Pooling kernel size	7
	Normalization type	Batch
	Activation type	None
Block 2	Convolution output channels	32
	Convolution kernel size	15
	Pooling type	Average
	Pooling kernel size	7
	Normalization type	Batch
	Activation type	Relu

configuration, which potentially may lead them to resort to using larger and more complex deep learning models and to incorporate specific fine-tuning strategies tailored to each sensor type in order to achieve satisfactory performance.

Our RLNAS framework alleviates this challenge by enabling the automatic selection of a suitable autoencoder for anomaly detection, thus simplifying the model selection process and allowing the effective use of simple deep learning models such as the autoencoder.

2.3.9 Application Example: Optimization of an Occupancy Estimation Model

Occupancy estimation in buildings involves the utilization of IoT sensors in conjunction with ML algorithms to determine the number of people present in different areas or rooms within a building. it plays an important role in transforming traditional buildings into smart, adaptable, and resource-efficient spaces. Indeed, This technology enables building managers and facility operators to gain real-time insights into occupancy patterns, helping them optimize building operations such as responsively adjusting heating, cooling, lighting, and ventilation systems based on the environmental conditions and actual occupancy levels, thereby reducing energy wastage. It also allows organizations to optimize the allocation of workspaces, leading to cost savings and improved productivity.

Anomalies in sensor data, stemming from factors like sensor malfunctions, hinder the ability of

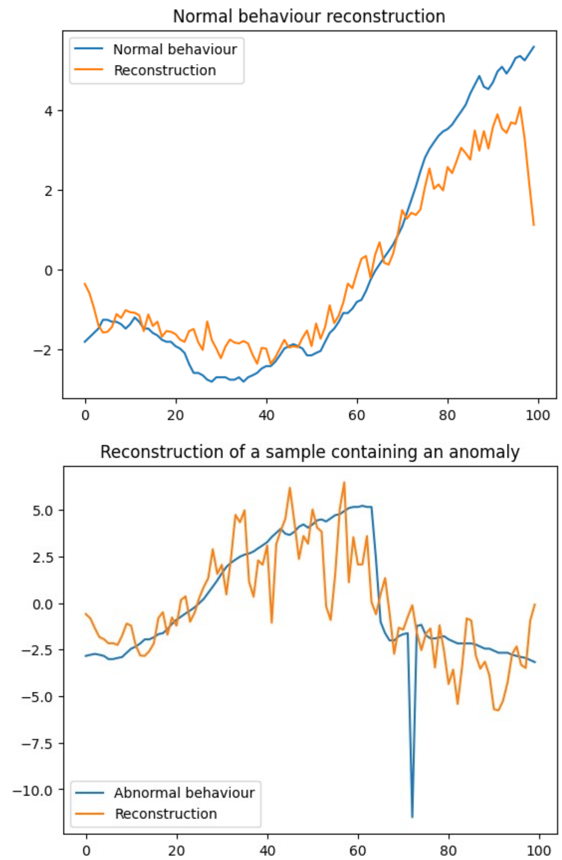


Figure 2.9: Reconstruction of a normal (top) and anomalous (bottom) data sample by the top discovered autoencoder architecture for the temperature measurements of the IWESTM40 weather station

ML models to learn generalizable patterns from sensor observations, therefore compromising the accuracy of occupancy estimation systems in smart buildings and leading to suboptimal resource allocation and energy efficiency. Therefore, a potential contribution of anomaly detection techniques like ours reside in enhancing the performance of such systems by detecting and addressing anomalies to ensures that ML models are trained using consistent, anomaly-free data.

To model this situation, we again use the office room dataset from [8] as it was originally created for the purpose of occupancy estimation. We first conduct a benchmark of several ML models using their default hyperparameters including Decision Tree, Random Forest, Gaussian Naïve Bayes, K Nearest Neighbors and Perceptron on the binary classification task of detecting whether the office room is occupied or not using anomaly-free measurements of temperature, humidity and power usage. The obtained performance is displayed in Table 2.5. Consequently, we opt to use the Perceptron model for this simulation, as it offered the best performance.

Then, we synthesize anomalous data points for the considered sensors using the anomaly generation approach that we mentioned earlier and inject them into regular sensor measurements to contaminate 3% of total data. This contaminated data is used to train the Perceptron model on occupancy estimation. This approach allows us to emulate the real-world scenario wherein anomalies often go unnoticed due to their infrequency and, hence, become embedded within the regular sensor data and continue to propagate into subsequent applications, such as occupancy estimation.

Following this, we generate new training data for our RLNAS framework and employ it to find a suitable autoencoder configuration for each of the sensors' data. These autoencoders are then used to detect and filter anomalies from the data used to train the occupancy estimation model. Our autoencoder technique focuses solely on identifying whether a time window contains an anomaly or not, whereas in our case, we want to filter the anomalous data points individually while retaining the regular data within the sequence to maximize the available occupancy training data. To achieve this, we flag a single anomalous point per time window, which corresponds to the highest distance between the original and reconstructed sequences.

Once this filtering process is complete, we proceed to train the occupancy estimator model using this refined dataset. This allows us to assess the extent to which anomaly detection contributes to improving the classifier's performance. In Table 2.6, we compare the performance of the Perceptron classification model trained on contaminated data with a second one that is trained on the same data but after removing the detected anomalies (including false positives). We observe that anomalies in the training data of the occupancy estimator despite only representing 5% of data have a catastrophic effect on the model's performance as it declined from an 88.47% accuracy and a $F1$ score of 64.39% to a lower accuracy of 64.91% and a significantly lower $F1$ score of only 46.01% indicating that this new data led the classifier to learn to privilege correctly classifying the majority class corresponding to an empty room and which represent 75% of the occupancy data. However, the second model trained on filtered data yields a higher accuracy of 88.43% and a better $F1$ score of 64.02% indicating that this model is more accurate overall but also strikes a much better balance

Table 2.5: Performance comparison of simple Machine Learning classification models on occupancy estimation

Model	Accuracy	F1 score
Decision Tree	73.25	48.40
Random Forest	80.43	54.39
Gaussian Naïve Bayes	79.55	00.00
K-Nearest Neighbors	81.78	55.37
Perceptron	88.47	64.39

Table 2.6: Performance comparison of a Perceptron classifier trained on contaminated data (5% anomalies) with a second one trained on the same data after filtering anomalies using our technique

Train data	Accuracy	F1 score	Precision	Recall
Contaminated	64.91	46.01	33.53	73.15
Filtered	88.43	64.62	86.18	51.69

between detecting the majority and the minority class. Moreover, its performance is close to the best achievable performance corresponding to a Perceptron trained on anomaly-free data (Table 2.5).

This experiment demonstrates the critical effect anomalies may have on the performance of ML models, and highlights how our anomaly detection framework can be used to enhance their robustness and, in cases, to transform unusable models into accurate ones. Ultimately, this showcases the importance of integrating anomaly detection techniques like ours into ML systems that operate on time series data to ensure their accurate and reliable operation.

2.4 Discussion and Future Work

Through our RLNAS approach, we have successfully identified appropriate autoencoder architectures, yielding satisfactory performance for anomaly detection in different kinds of smart building sensor data.

Nevertheless, one limitation in our current NAS approach is that even though the RL agent can choose not to include layers, the maximum number of layers still has to be set prior to starting the search process as it a parameter of the size of the agent's neural network. This hyperparameter impacts the complexity and therefore the accuracy of the child model. Moreover, we cannot choose a value that is unnecessarily high, expecting the agent not to stack more layers when convenient as it badly affects the training time.

Moreover, the significant computational times required to conduct an RLNAS process may pose a barrier to its adoption in some situations and even though techniques like weight sharing could be used to avoid training each sampled architecture from scratch [90], a prior study has demonstrated that this will introduce a bias in the search process, as it increases the likelihood of sampling architectures with better initial performance and similar structures [28]. Hence, an interesting future research direction would be to evaluate how to computationally optimize RLNAS without sacrificing its efficiency.

Besides, for the anomaly detection method we used, a limitation we observed through the analysis of different architectures proposed by the RL agent is that as the autoencoder is presented with more diverse and complex signal patterns, it might struggle to learn to generalize its reconstruction to all kinds of sequences, or it might also learn to reconstruct all patterns, even anomalous ones, as it fails to detect and learn a single behavior pattern. This explains the relatively significant false negative rate we achieved. This is why, this proposed approach works best when the normal signal patterns are either highly seasonal or have low overall variation around a mean value, like for the case of weather data.

One particular case is when input sequences include patterns that might seem uncorrelated when we observe time windows individually, but that might express daily or weekly seasonality when observed over a longer time period. Therefore, in a future work, we aim to enhance our method by exploiting such seasonal patterns through more complex feature engineering, which will also allow us to tackle more diverse kinds of sensors.

Additionally, though we explicitly chose to use the 99^{th} percentile of the training reconstruction loss as an anomaly threshold to ensure high precision, in cases where false positives are tolerable or preferable to false negatives, it would be interesting to consider other anomaly definitions or even to consider the threshold as a hyperparameter and search for the value that would strike the best balance between precision and recall. Moreover, in this research, we make the assumption that there is access to clean data for training the Autoencoder models. However, it's important to acknowledge that such clean data may not always be readily available. Therefore, in our future research, we plan to focus on the development of anomaly detection frameworks in more challenging, real-world, noisy environments.

Furthermore, the scope of this study was primarily restricted to outliers that manifest as isolated data points. And while this represents a major category of anomalies, other forms may also be observed in real life scenarios, particularly pertaining to sensor and system failures, such as signal shape anomalies which concern for instance a sudden change in the mean value, in the standard deviation or in the slope of future measurements. Investigating such concerning anomalies will also be a subject of future research.

Ultimately, we ought to explore additional applications of Reinforcement Learning and anomaly detection algorithms within the context of smart buildings because these techniques have already demonstrated promising results in enhancing control mechanisms and identifying anomalies. Yet, delving deeper into their potential applications may reveal more unexplored possibilities. For instance, by incorporating RL algorithms, smart buildings could optimize various processes, such as climate control, resource allocation, and occupant comfort, leading to substantial energy savings and cost efficiencies. Moreover, anomaly detection techniques can play a key role in identifying and addressing irregularities in building operations, ensuring early detection of faults, potential hazards, or security breaches. As the potential benefits of these technologies extend beyond the current scope, further research and experimentation are necessary to fully exploit their capabilities, making our buildings smarter, more sustainable, and safer for both occupants and the environment, therefore creating a positive impact on the future of smart infrastructure.

2.5 Conclusion

In this research, we leveraged a reconstruction based anomaly detection method using autoencoders to detect anomalies in univariate time series data of smart building sensors. To overcome the complexity of choosing the best configuration for this model, we resorted to a neural architecture search technique that trains a reinforcement learning agent to explore a hyperparameter search space and find the best configuration through trial and error. We test this approach on our custom anomaly detection dataset and achieve competitive results. In addition, we showcase an illustrative case where this method is used to improve the performance of an occupancy estimation machine learning model by filtering anomalies from its training data. Moreover, the analysis of agent choices shows that the autoencoder configurations found may not be immediately apparent or intuitive to a developer, which highlights the effectiveness of this approach. In this way, we demonstrate that RLNAS can make for the lack of domain and task-specific knowledge that might be required to design an effective neural network.

The aim of this system is to identify abnormal behaviors and deviations in temporal data collected from IoT sensors in smart buildings. This proposed techniques will enable the detection of anomalies related to power consumption, control system failures, and sensor malfunctions, thus facilitating proactive maintenance, optimizing energy consumption, and improving operational efficiency.

Chapter 3

Unsupervised Anomaly Detection and Imputation in Noisy Time Series Data for Enhancing Load Forecasting

3.1 Introduction

Despite the remarkable recent machine learning led advancements in time series forecasting [61], it is evident that merely developing more accurate models is not sufficient, as the effectiveness of these models is greatly reliant on the quality of available historic data [87] to be used for training these models. In fact, anomalies characterized by sudden deviations or irregular patterns in power load data, obscure the underlying trends in the time series data hence disrupting the predictive capabilities of machine learning algorithms [99]. Consequently, the reliability and precision of load forecasts are compromised, leading to suboptimal decision-making and misallocation of resources.

While numerous techniques for anomaly detection in time series data exist, many of them (such as reconstruction-based or distance based methods) assume the availability of clean, anomaly-free data for training or require anomalies to be accurately labeled in the historic data (e.g. supervised machine learning models), which isn't practical in real-world scenarios. Furthermore, simply detecting anomalies isn't sufficient, as omitting these values could compromise the effectiveness of forecasting by generating abrupt load patterns. We must instead replace these patterns with new plausible ones. This is not a trivial task as traditional imputation methods, such as replacing with a day's mean value or by the next day's value, do not work well for long sequences of anomalies as they result in disrupt patterns. Denoising autoencoders have emerged as a promising alternative; thus, we propose developing a model that integrates denoising autoencoders with our anomaly detection system to create a data-cleaning framework that aims to make anomalous data suitable for training models on downstream tasks like forecasting. This model is also benchmarked against KNN imputation, another effective approach identified in the literature. In this work, rather than introducing a new accurate forecasting model for noisy and anomaly-infected datasets, we want to quantify how the combined anomaly detection and imputation preprocessing pipeline boosts the forecasting accuracy of existing models. Because traditional statistical models and advanced neural networks like transformers show limitations with long sequences and temporal information loss, for our experiments, we select two models. The first one is an RNN-based model that we chose because of the widespread use of this type of forecasting model. The second model is selected because it achieves state-of-the-art on several datasets.

In sum, this research addresses the research gaps by proposing a memory-based framework for anomaly detection in time series data. Unlike traditional methods that require labeled data, our approach is unsupervised, learning predominant patterns from the data and identifying anomalies as patterns that significantly deviate from the features observed during training and stored in memory. It also overcomes the need for clean data by detecting and discarding irregular patterns from the memory bank during the training phase, making this method particularly suitable for both noisy and anomaly-infected time series data, as it is able to mitigate the impact of these data quality issues. The detected anomalies are replaced by plausible patterns using an RNN based imputation model trained on reconstructing data deemed anomaly-free after iterative and random manual corruption. Through a rigorous evaluation across data from different kinds of buildings, we demonstrate the effectiveness of our proposed approach in identifying and rectifying anomalies to enhance the usability of the data for forecasting. This is achieved by quantifying the forecasting accuracy gained by using this method through a comparison between the performance of forecasting models trained on anomaly-contaminated data and those trained on the same data after detecting and imputing anomalies.

Ultimately, this research contributes to the advancement of energy management practices by enabling more accurate and reliable load forecasting for electrical grids and smart buildings, paving the way towards sustainable and efficient energy utilization.

The rest of this chapter is structured as follows: In section 3.2 we offer a detailed explanation of our proposed system including the anomaly detection framework, anomaly imputation system and the forecasting models used for evaluation. In section 3.3, we further explain how these components are interconnected during our experiments and provide details into our specific methodology, implementation and evaluation approach. Following this, in section 3.4, results are presented and explained, offering insights into the performance of our system across various datasets and scenarios, and finally section 3.5 engages in a discussion of our system, the obtained results, and outlines potential avenues for future research and development.

3.2 Proposed System

In this section, we introduce and theoretically explain the distinct components of our framework, including anomaly detection, anomaly imputation, and load forecasting.

3.2.1 Anomaly Detection

Overview

This anomaly detection methodology relies on a memory bank to store reference features extracted during the training phase and subsequently employing them during inference to compute an anomaly score.

Initially, the memory bank is populated with feature extracted from training data using a frozen backbone. In this process, we extract patch features (i.e. local descriptors of a small region of the signal) as they are able to capture both single-point anomalies and anomalous sequences. Plus, they encapsulate temporal relationships within time series data, effectively describing each data sample in relation to its adjacent values. Subsequently, the memory bank is filtered to remove anomalous features, ensuring that only anomaly-free features remain for computing anomaly scores during inference. These scores are determined by measuring the distance between a test sequence and its most analogous memory-stored pattern, within the feature space, where anomalous patterns are easier to distinguish from regular ones.

The main components are further discussed below, and a general high-level algorithm is provided in Algorithm 2 to summarize the memory bank's construction process.

Algorithm 2 Constructing an Anomaly-Free, Patch-Level Feature Memory Bank

Input: Time series sequences $\mathcal{X}_N$, backbone Φ, hierarchy j, random linear projection ψ, denoising threshold τ, new coreset size k.
Output: Memory bank $\mathcal{M}_c$.

$\mathcal{M} \leftarrow \{\}$ ▷ Populate the memory bank.
for $x_i \in \mathcal{X}_N$ **do**
    Compute locally aware patch features $P_i = P(\Phi_j(\mathbf{x}^i))$ (Eq. 8)
    $\mathcal{M} \leftarrow \mathcal{M} \cup P_i$
end for
Build an MG Distribution $\mathcal{N}(\mu, \Sigma)$ (Eq. 11) ▷ Denoise the memory bank.
for $m_i \in \mathcal{M}$ **do**
    Compute and store an anomaly weight W_{m_i} (Eq. 12)
end for
Discard elements m_i corresponding to top $\tau\%$ highest anomaly weights W_{m_i} from $\mathcal{M}$
$\mathcal{M}_c \leftarrow \{\}$ ▷ Coreset Reduction.
for $i \in [1, \ldots, k]$ **do**
    Calculate new memory elements m_i (Eq. 14)
    $\mathcal{M}_c \leftarrow \mathcal{M}_c \cup m_i$
end for

Patch Feature Extraction

During this first phase of feature extraction, a generic pre-trained backbone neural network denoted as Φ (e.g. ResNet) is used to extract patch features from a given time series sequence $\mathbf{x}^i \in \mathbb{R}^{l \times d}$ belonging to a collection of subsequences extracted from a main time series $T \in \mathbb{R}^{L \times d}$ of size L and dimensionality d through a sliding window process of stride δ, i.e. :

$$\mathbf{x}^i \in \mathcal{X}_N = \{T_{0:l,\ldots}, T_{\delta:\delta+l,\ldots}, \ldots, T_{(N-1)\delta:(N-1)\delta+l,\ldots}\} \tag{7}$$

where $i \in \{0, \ldots, N\}$ with $N = \lfloor \frac{L-l}{s} \rfloor + 1$ the number of time windows of size l.

Following conventional notations, the features extracted at the hierarchy level j of the backbone for the sequence $\mathbf{x}^i$ are denoted as $\Phi_{i,j} = \Phi_j(\mathbf{x}^i) \in \mathbb{R}^{c \times l \times d}$.

While this network is not specifically trained on a relevant time series task, we posit that a considerable discrepancy still exist between features extracted from a regular sequence and those extracted from an anomalous one. Notably, we focus on low and middle-level features instead of high-level features that capture task-specific representations that are relevant to the backbone's original training task. However, while these features are more general, they possess a smaller awareness of their local neighborhood due to a reduced receptive field, potentially missing relevant temporal context information for characterizing anomalous data samples in relation to neighboring observations.

To remedy to this, locally aware patch features $\mathcal{P}(\Phi_{i,j})$ were proposed in [94], aggregating feature vectors from the neighboring region of size k of the data point (l, d) denoted $\mathcal{N}_k^{(l,d)}$. They are computed as follows:

$$\mathcal{P}(\Phi_{i,j})^{(l,d)} = \Phi_{i,j}(\mathcal{N}_p^{(l,d)}) = f_{agg}(\Phi_{i,j}(a,b) \mid (a,b) \in \mathcal{N}_p^{l,d}) \tag{8}$$

where f_{agg} is a defined aggregation function (e.g. average pooling) and the neighborhood is defined as:

$$\begin{aligned} \mathcal{N}_k^{(l,d)} = \{(a,b) \ |a \in [l-[\frac{k}{2}],...,l+[\frac{k}{2}]], \\ b \in [d-[\frac{k}{2}],...,d+[\frac{k}{2}]]\} \end{aligned} \tag{9}$$

Finally, a memory bank $\mathcal{M}$ is populated with the locally aware feature maps generated for each time series sequence $x_i, i \in \{1, \dots, N\}$ observed in the training phase as follows:

$$\mathcal{M} = \bigcup_{i=0}^{N} P(\Phi_j(\mathbf{x}^i)) \tag{10}$$

Noise Discrimination

$\mathcal{M}$ serves as the basis for computing anomaly scores through proximity to the most similar stored features. However, this paradigm assumes that the training data is anomaly free, as in cases where the training data contains anomalies, an anomalous test sequence may have similar features stored in $\mathcal{M}$, resulting in a low anomaly score and leading to a false negative classification.

Therefore, and similarly to [59], we incorporate a noise discrimination mechanism to filter anomalous patches from the memory bank. This assumes that regular feature corresponding to the predominant anomaly-free patterns share a similar distribution in the feature space, while anomalies, which are relatively rare events with considerably different patterns, have larger variance. Therefore, normal features are in a dense feature space while anomalous patterns are sparse and can be identified using the Mahalanobis distance, which computes a distance between a point and a distribution.

To this end, a Multi-variate Gaussian (MG) Distribution $\mathcal{N}(\mu_{l,d}, \Sigma_{l,d})$ is constructed, with $\mu_{l,d}$ as the batch mean of $\Phi_i(l,d)$ and the sample covariance defined as:

$$\Sigma_{l,d} = \frac{1}{n-1}\sum_{n=0}^{N}(\Phi_n(l,d) - \mu_{l,d})(\Phi_n(l,d) - \mu_{l,d})^T + \epsilon I \tag{11}$$

The ϵI term ensures that the matrix is full rank and invertible [34].

Subsequently, the Mahalanobis distance is computed for the elements $m_i \in \mathcal{M}$ as follows and serve to identify anomalies in the memory bank as features with high distance.

$$W_{m_i}(l,d) = \sqrt{(\Phi_{m_i}(l,d) - \mu_{l,d})^T \Sigma_{l,d}^{-1} (\Phi_{m_i}(l,d) - \mu_{l,d})} \tag{12}$$

This distance is computed for all samples in $\mathcal{M}$, and those corresponding to the $\tau\%$ highest distances are removed, where τ represents a memory bank denoising threshold parameter. Moreover, this distance is also used during inference to scale the anomaly score of test features according to the abnormality level of the stored features to further account for noisy patterns even if they are not filtered from $\mathcal{M}$.

Coreset Reduction

A notable challenge in memory-based anomaly detection approaches arises from the storage demands associated with larger datasets, where the length of T leads to a substantial number of saved features in $\mathcal{M}$. This necessitates considerable storage space and can result in slower inference times. Coreset selection offers a solution to this issue by identifying a subset $\mathcal{M}_c \subset \mathcal{M}$ that, despite being smaller, closely approximates solutions over $\mathcal{M}$, thereby reducing inference time while preserving performance. Given that this approach relies on nearest neighbor calculations, we employ the minmax facility location technique, as suggested in prior research [94, 101]:

$$\mathcal{M}_c = \underset{\mathcal{M}_c \subset \mathcal{M}}{\text{argmin}}(\max_{m \in \mathcal{M}}(\min_{n \in \mathcal{M}_c} \|m - n\|_2)) \tag{13}$$

This calculation is performed using a greedy approximation, similar to the methodology outlined in [101]. Additionally, random linear projections are applied to reduce the dimensionality of the memory-stored vectors $m \in \mathcal{M}$: $\psi : \mathbb{R}^d \to \mathbb{R}^{d^*}$ where $d < d^*$ [103].

Consequently, we initialize an initially empty reduced set $\mathcal{M}_c$ and iteratively calculate new elements as displayed in Equation 14, repeating the calculation until the target coreset size is reached:

$$m_i = \underset{m \in \mathcal{M} - \mathcal{M}_c}{\text{argmax}}(\min_{n \in \mathcal{M}_c} \|\psi(m) - \psi(n)\|_2) \tag{14}$$

Model Inference

During inference, an anomaly score is calculated for an unseen time series test sequence x^{test} by calculating its patch features (in the same way as for the elements in $\mathcal{M}$) then by determining the nearest neighbor of each patch $p \in \mathcal{P}(\Phi_j(x^{\text{test}}))$ as follows:

$$m^* = \underset{m \in \mathcal{M}}{\text{argmin}} \|p - m\|_2 \tag{15}$$

For each patch, an anomaly score is then computed as a distance to its nearest neighbor. This distance is scaled by the weight W_{m^*}, obtained during the noise discrimination phase, to account for the abnormality level of the memory-stored features:

$$s_p = W_{m^*} \|p - m^*\|_2 \tag{16}$$

Setting a threshold on these anomaly scores (e.g., using a percentile value if the data contamination rate is estimable) allows for the identification of anomalous patches and by examining a patch's receptive field, timestamps corresponding to anomalies can be detected, as illustrated in Fig. 3.4.

Moreover, an anomaly score for the entire sequence (of size l) may be computed through the application of an aggregation function over the patch scores (e.g., average or max).

To summarize, this approach solves the identified literature issues by implementing an unsupervised learning method, eliminating the need for labeled data. Instead, it can work with unlabeled data to identify the underlying patterns governing the time series. Additionally, it overcomes the challenge of requiring clean data for effective training through the memory bank denoising mechanism. This filters anomalous samples, only keeping clean ones, allowing the model to learn the regular pattern and later discriminate regular from anomalous patterns.

3.2.2 Anomaly Imputation

Anomaly Imputation or missing data imputation in general can be framed as a reconstruction problem, where the goal is to generate a new complete time series that aligns closely with the existing observed values while also accurately deriving missing values based on the patterns observed in the existing data, ensuring coherence of the time series variation and fidelity to its underlying structure [137].

Autoencoders, particularly recurrent denoising autoencoders (RDAEs), are well suited for reconstructing incomplete time series data. During training, the model learns to fill in missing values by encoding the input into a lower-dimensional latent space and then decoding it back to its original dimensionality, with the aim of minimizing the reconstruction error. Unlike traditional autoencoder architectures that focus on learning a compact representation of static inputs, RDAEs are tailored to handle sequential data by incorporating recurrent neural network (RNN) layers into their structure that help capture temporal patterns and dependencies.

In fact, to fully capitalize on this temporal correlation, we employ the bidirectional Long Short Term Memory (bi-LSTM) model in the internal structure of the autoencoder. This is an RNN architecture that utilize two LSTM networks in a bidirectional configuration to capture both past and future context in sequential data by processing the input sequence in two directions: forward and backward, resulting in two distinct latent feature representation that, similarly to [137], we concatenate as following

$$
\begin{aligned}
h &= enc(\mathbf{x}) \\
&= \overrightarrow{\text{LSTM}}(x_1, x_2, \ldots, x_l) \;\oplus\; \overleftarrow{\text{LSTM}}(x_l, x_{l-1}, \ldots, x_1)
\end{aligned} \tag{17}
$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_l)$ denotes the input sequence of length l and $\oplus$ denotes the concatenation operation.

This choice is made because our task involves imputing data sequences identified as anomalous from a large historical dataset that will subsequently be used to train forecasting models. Hence, for any given time step to be imputed, we have access to both past and future values. Therefore, processing the data in both directions allows the model to capture dependencies from both past and future patterns, providing a more comprehensive understanding of the sequence.

LSTM in itself is an RNN architecture designed to overcome the vanishing gradient problem and capture long-range dependencies in sequential data. Unlike traditional RNNs, LSTMs incorporate gated mechanisms to regulate the flow of information through the network [113].

At each time step t, an LSTM unit computes three main components: the input gate i_t, the forget gate f_t, and the output gate o_t. These gates control the flow of information into and out of the LSTM unit, allowing it to selectively remember or forget information over time.

The input gate i_t determines how much new information should be added to the cell state c_t based on the current input x_t and the previous hidden state h_{t-1}. Similarly, the forget gate f_t regulates the extent to which the previous cell state c_{t-1} should be retained. Finally, the output gate o_t determines the amount of information that should be output as the hidden state h_t.

The cell state c_t acts as a memory unit that can store information over long periods, allowing LSTMs to capture long-term dependencies in the input sequence. The hidden state h_t is computed based on the cell state and the output gate, providing a compact representation of the input at each time step.

The equations governing the computation of the LSTM gates and states are as follows:

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \tag{18}$$

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \tag{19}$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \tag{20}$$

$$g_t = \tanh(W_{gx}x_t + W_{gh}h_{t-1} + b_g) \tag{21}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{22}$$

$$h_t = o_t \odot \tanh(c_t) \tag{23}$$

where σ denotes the sigmoid activation function, $\tanh$ denotes the hyperbolic tangent activation function, c are weight matrices for the input gate, forget gate, output gate, and cell state, respectively, and W_{ih}, W_{fh}, W_{oh}, W_{gh} are weight matrices for the corresponding gates and states from the previous hidden state. Additionally, b_i, b_f, b_o, b_g represent the bias terms for the gates, and $\odot$ denotes element-wise multiplication.

In our framework, let $x \in \mathbb{R}^l$ represents the input time series sequence, with l denoting its size (acquired via a sliding time window). During inference, we utilize a binary mask object $\mathbf{m} \in [0, 1]^l$ to store the indices of detected anomalies. Here, m_t at time step t indicates whether x_t is classified as an anomaly:

$$m_t = \begin{cases} 0 & \text{if } x_t \text{ is an anomaly.} \\ 1, & \text{if } x_t \text{ is a regular observation.} \end{cases} \tag{24}$$

Subsequently, we perform element-wise multiplication between the input sequence $\mathbf{x}$ and the anomaly mask $\mathbf{m}$, denoted as $\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{m}$, before passing it to the RDAE model. The resulting decoded sequence, $\hat{\mathbf{x}} = dec(enc(\tilde{\mathbf{x}}))$, approximates a complete, anomaly-free version of the input sequence where the decoder is a bi-LSTM layer followed by a Fully-Connected (FC) layer. This sequence serves as a replacement for the missing values, yielding a new imputed sequence through the formula:

$$\mathbf{x}_{\text{imputed}} = \mathbf{x} \odot \mathbf{m} + \hat{\mathbf{x}} \odot \bar{\mathbf{m}} \tag{25}$$

where $\bar{\mathbf{m}}$ represents the complementary mask to $\mathbf{m}$.

The model architecture is illustrated in Fig. 3.1.

During the neural network's training phase, we utilize the data sequences deemed anomaly-free. For each sample, we generate a random mask $\mathbf{m}_r$ where segments of values are randomly set to zeros. To train the network, we employ a loss function that quantifies the disparity between the original complete sequences and the recovered sequences after omitting data patches (Eq. 26). This approach emulates the scenario of detecting an anomalous patch of data, masking it, and then using the Autoencoder to fill the unknown masked sequence with plausible values. The RDAE learns to fill missing values with plausible ones by iteratively masking patches at random positions. This allows it to learn to generate a complete representation of the input time series while capturing the underlying patterns and the possible values for any given position (a patch starting at any hour). This enables the model to effectively impute both individual data points and entire sequences. Moreover, in our experiments, the model is trained over multiple epochs, with the random mask positions changing between epochs. This variation mimics being exposed to new previously unseen data and hence enhances the model generalizability to a wider set of anomalies. We use the Early Stopping algorithm to terminate training when the model's performance stabilizes (i.e., when improvements on the validation set plateau).

$$\begin{aligned}\mathcal{L} &= \text{MSE}(\mathbf{x}, \hat{\mathbf{x}}) \\ &= \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \quad = \frac{1}{n}\sum_{i=1}^{n}[x_i - dec(enc(x \odot \mathbf{m}_r))_i]^2\end{aligned} \tag{26}$$

3.2.3 Load Forecasting

We aim to assess the extent to which detecting and imputing anomalies may enhance the accuracy of load forecasting models. To accomplish this, we initially train and evaluate forecasting models using raw anomaly-contaminated data. Subsequently, we compare its performance to that achieved under identical conditions but after imputing the detected anomalies. To offer a comprehensive understanding of this performance enhancement, we conduct experiments using two distinct forecasting models: the simple, well-established and widely-used RNN-based Seq2seq model and

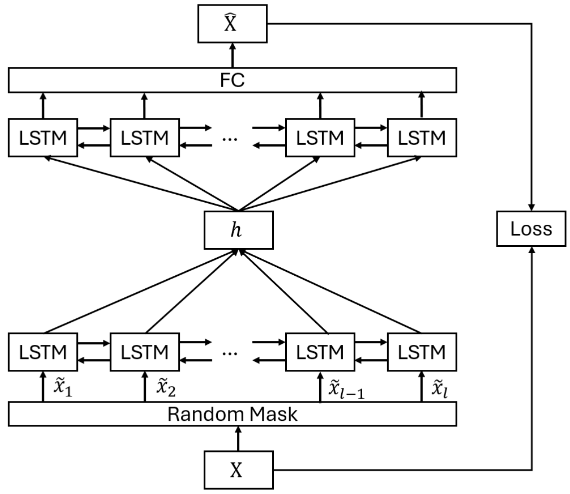


Figure 3.1: Illustrative diagram of the imputation model

the cutting-edge SCINet model from [79] that outperforms various state-of-the-art models on multiple benchmarks.

Seq2seq Model

The idea behind this model is to frame the time series forecasting problem as a sequence-to-sequence generation framework, employing Gated Recurrent Units (GRUs) to adeptly capture temporal dependencies. This architecture is composed of a recurrent encoder and a recurrent decoder. The encoder sequentially processes input time series observations, leveraging interconnected GRU cells to encode it into a context vector that encapsulates the temporal characteristics of the input sequence [25]. The overall architecture of this model is illustrated in Fig. 3.2.

In the following, let $(x_i)_{i=1}^{I}$ be the input sequence of size I. We want to forecast the next J timesteps, i.e. predicting $(x_j)_{j=0}^{J}$.

Initially, each input observation is passed through interconnected GRU cells, which iteratively compute hidden states. Formally, the hidden state h_i^{enc} at time step i is computed as follows:

$$h_i^{\text{enc}} = enc(x_i, h_{i-1}^{\text{enc}}) \tag{27}$$

The final hidden state h_I^{enc} serves as the context vector summarizing the entire input sequence.

In contrast, the recurrent decoder aims to generate the output time series based on the context vector produced by the encoder. Initially, the context vector initializes the decoder's hidden state $h_0^{\text{dec}} = h_I^{\text{enc}}$. For each time step in the forecast horizon j in the output sequence, the decoder's hidden state h_j^{dec} is computed using the previous output y_{j-1} and the preceding hidden state h_{j-1}^{dec}. This process is mathematically represented as follows

$$h_j^{\text{dec}} = dec(y_{j-1}, h_{j-1}^{\text{dec}}) \tag{28}$$

Following the computation of the decoder's hidden states, the output at each time step is generated utilizing a fully connected layer with a softmax activation function, thereby yielding the predicted value $\hat{y}_j$. This can be expressed as:

$$\hat{y}_j = \text{softmax}(W h_j^{\text{dec}} + b) \tag{29}$$

where W is the weight matrix, b is the bias vector, and h_j^{dec} is the hidden state of the decoder at time step j.

Similar to LSTM, GRU incorporates gated mechanisms to regulate the flow of information within the network [29]. At each time step t, a GRU unit computes two main components: the update gate z_t and the reset gate r_t. These gates control the flow of information, allowing GRU to selectively update its hidden state. The update gate z_t determines how much of the previous hidden state h_{t-1} should be preserved and how much of the new candidate activation $\tilde{h}_t$ should be incorporated. On the other hand, the reset gate r_t decides how much of the previous hidden state h_{t-1} should be ignored when computing the candidate activation $\tilde{h}_t$.

The candidate activation $\tilde{h}_t$ is computed based on the current input x_t and the previous hidden state h_{t-1}, capturing the information relevant for the current time step. The hidden state h_t is then updated as a combination of the previous hidden state and the candidate activation, weighted by the update gate.

The equations governing the computation of the GRU gates and states are expressed as follows,

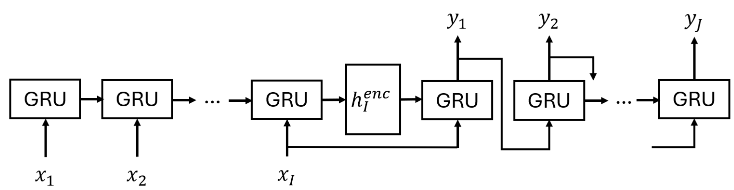


Figure 3.2: Illustrative diagram of the Seq2seq forecasting model

using similar notations to the ones previously used for LSTM:

$$z_t = \sigma(W_{zx}x_t + W_{zh}h_{t-1} + b_z) \tag{30}$$

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_{t-1} + b_r) \tag{31}$$

$$\tilde{h}_t = \tanh(W_{hx}x_t + r_t \odot (W_{hh}h_{t-1}) + b_h) \tag{32}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{33}$$

Through iterative training using the Mean Squared Error (MSE) loss function, the model optimizes its parameters, minimizing the disparity between predicted and actual output sequences.

The main advantage of GRUs over LSTM lies in their simpler architecture, which translates to easier training and reduced need for parameter optimization, enabling us to conduct experiments across a wide range of data types more efficiently.

SCINet model

SCINet adopts a recursive downsample-convolve-interact approach within an encoder-decoder framework [79].

The encoder component comprises SCI-Blocks, which first downsample the input data into two sub-sequences that each undergoes processing via a series of convolutional filters, extracting distinct temporal features from each part. Mathematically, this involves splitting a set of features $F = (x_i)_{i=1}^{I}$ into two sub-sequences by segregating observations at odd and even positions, denoted as F_{even} and F_{odd} respectively, thus reducing temporal resolution while retaining most of the information.

To compensate for downsampling-caused information loss. An interactive learning framework is proposed. This framework employs four distinct 1D convolutional neural networks with learnable weights ϕ, ψ, ρ and η to project F_{even} and F_{odd} into hidden states where the following scaling transformations are applied (Eq. 37) to allow information exchange between the two sub-sequences and making the receptive field at each layer larger.

$$F^s_{\text{odd}} = F_{\text{odd}} \odot \exp(\phi(F_{\text{even}})) \tag{34}$$

$$F^s_{\text{even}} = F_{\text{even}} \odot \exp(\psi(F_{\text{odd}})) \tag{35}$$

$$F'_{\text{odd}} = F^s_{\text{odd}} \pm \rho(F^s_{\text{even}}) \tag{36}$$

$$F'_{\text{even}} = F^s_{\text{even}} \pm \eta(F^s_{\text{odd}}) \tag{37}$$

SCINet organizes multiple SCI-Blocks into a binary tree structure, where the two outputs produced by each block (F'_{even} and F'_{odd}) serve each as a separate input for a subsequent SCI-block. This arrangement enables the model to capture rich features from multiple resolutions, representing both local and global perspectives of the time series, effectively modeling time series with complex temporal dynamics.

In the final stage, a new sequence with enhanced predictability is constructed by reversing the odd-even splitting operation and concatenating them with the original time series for forecasting using a fully-connected network as the decoder.

This architecture learns to accurately forecast time series data through minimization of the MSE loss between the ground truth horizon sequence and the decoded sequence.

3.3 Experimental Framework

In this section, we begin by introducing the data sources and anomaly types we considered in our experiments. We then illustrate the interconnections between the various components outlined in the previous section and how they are employed. We also expose the implementation-specific details of our experiments. Following this, we present baselines we employ for comparison, followed by the metrics we employ for evaluation purposes.

3.3.1 Data

Datasets Description

To offer broad insights from our experiments and to demonstrate diverse applications of the proposed anomaly detection and imputation modules, we opt to utilize three distinct load datasets sourced from diverse settings, varying in building type, scale, and other factors. These datasets are as follows:

- The Australian Energy Market Operator (AEMO) dataset offers time series data on electricity demand aggregated for the states of Australia, with considerably different load patterns for each state. We collect data for the years 2015 to 2020, featuring a sampling rate of 30 minutes. This data is invaluable for forecasting electricity demand trends at the regional scale, aiding governments in the optimization of energy production, energy resource allocation and infrastructure planning.
- The industrial Park dataset from [143] provides anomaly-free electric power load data for various building types in an industrial park in Suzhou, China (Commercial, Office, Public and Residential buildings). It includes 6 years of data spanning 2016 to 2021, captured with a 30-minute time resolution. This dataset is chosen to evaluate the system's generalizability across various building types and settings.
- The Predis-MHI dataset reports the electricity consumption of a tertiary living lab building located in the GreEn-ER building in Grenoble [83]. Observations are recorded with a 1-hour time resolution, covering the years 2016 to 2022. Notably, this dataset already encompasses multiple anomalies; hence, we aim to detect and impute these anomalies so we do not synthesize new ones.

Synthetic Anomalies Generation

Given that, as of our current knowledge, there is no publically available dataset specific to time series load data that contain genuine anomalies, we resort to creating our own anomalous data by injecting artificial anomalies into regular data.

In order to generate diverse synthetic load data specific anomalies with real-world properties, we refer to the study conducted in [111] where authors analyze the dataset that was originally curated in [117] which include records of real energy and power time series data for 600 smart meters installed in different locations at the Karlsruhe Institute of Technology (KIT). They identify the unusual patterns and define four types of commonly occurring load anomalies caused by technical faults.

Anomaly Type 1 For this specific anomaly type, the energy time series exhibits a sudden drop to zero, followed by a subsequent rebound to a plausible value after a time period denoted by τ. In the power time series, this manifests as a negative spike, a sequence of zero values, and then a positive spike, effectively restoring the energy to a reasonable level. Such behavior is indicative of a potential issue in the recording of values by the energy meter.

In the following, let $E = e_1, e_2, \ldots, e_N$ represents the energy time series, and $P = p_1, p_2, \ldots, p_N$ denotes the corresponding power time series, k an offset parameter and $u \sim \mathcal{U}_{[0,1)}$.

An anomaly happening at time step i can be accurately modeled using the following equations:

$$e_{i+t} = 0, \quad 0 \leq t < \tau \tag{38}$$

$$p_{i+t} = \begin{cases} k - (\sum_{j=1}^{i-1} p_j), & t = 0 \\ 0, & 0 < t < \tau - 1 \\ k + \sum_{j=1}^{i+\tau-1} p_j & t = \tau - 1 \end{cases} \tag{39}$$

Anomaly Type 2 The second anomaly type describes situations where the energy gradient decreases, stagnates for a duration τ, then increases gradually to reach a correct value. In the power domain, this translates to a decline to zero values, followed by a positive spike corresponding to the power accumulated during τ. This anomaly arises from disruptions in the transmission of sensor measurements and is effectively synthesized through the following equations:

$$e_{i+t} = u \cdot e_i + (1 - u) \cdot e_{i-1}, \quad 0 \leq t < \tau \tag{40}$$

$$p_{i+t} = \begin{cases} u \cdot p_i, & t = 0 \\ 0, & 0 < t < \tau - 1 \\ (1-u) \cdot p_i + \sum_{j=i+1}^{i+\tau-1} p_j & t = \tau - 1 \end{cases} \tag{41}$$

Anomaly Type 3 This anomaly manifests as a sudden decrease in the energy time series, resulting in a negative spike in the power domain. Such behavior is often induced by an external adjustment of the sensor, such as recalibration to align readings from multiple smart meters. This anomaly is formulated as follows

$$e_{i+t} = e_{i+t} - u \cdot |e_i - e_{i-1}|, \quad t \geq 0 \tag{42}$$

$$p_i = -u \cdot p_{i-1} \tag{43}$$

In other identified cases the power spike is significantly higher and corresponds to a reset of the smart meter as follows.

$$e_{i+t} = e_{i+t} - e_i \quad t \geq 0 \tag{44}$$

$$p_i = -\sum_{j=1}^{i-1} p_k - k \tag{45}$$

Anomaly Type 4 This anomaly type is distinguished by a sudden surge in the energy gradient, leading to a positive power spike. External adjustments to the smart meter can often induce this behavior. The synthesis of these anomalies closely parallels the previous anomaly type but involves a distinct sampling interval for u, reflecting the estimated spike scale:

$$e_{i+t} = e_{i+t} + u \cdot |e_i - e_{i-1}|, \quad t \geq 0 \tag{46}$$

$$p_i = u \cdot p_{i-1} \tag{47}$$

An illustrative case is displayed in Fig. 3.3 showcasing the injection of each of the different anomaly type at random positions into the same regular energy and power measurements.

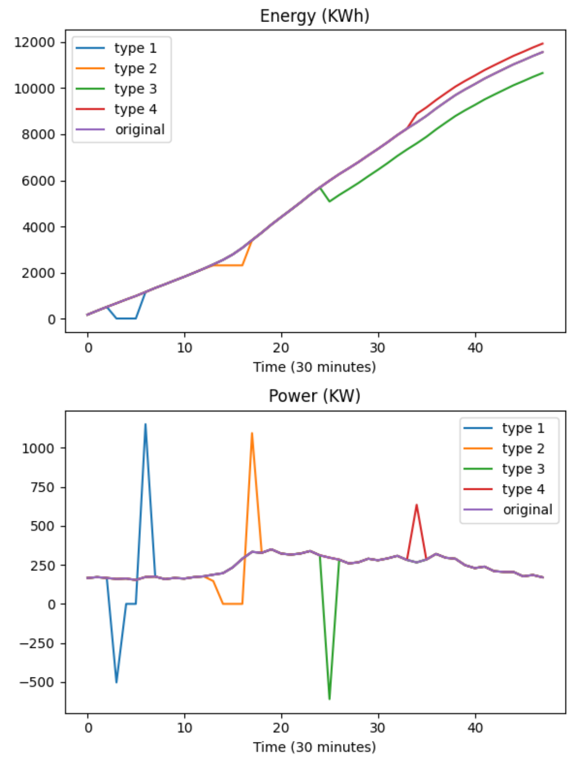


Figure 3.3: Synthetic energy and corresponding power anomalies generated for the load data of a commercial building from the Industrial Park dataset

3.3.2 Operational Setup

Experimentation Procedure

The initial phase involves data preparation, where anomalies are randomly generated across all types and injected into regular time series to achieve a data contamination rate of $p\%$, where anomalies are distributed among $d\%$ of days.

This dataset will be ultimately used to train and assessing the effectiveness of the forecasting, both pre- and post-anomaly detection and imputation. Hence, to ensure unbiased evaluation, we intentionally contaminate only an initial segment of the dataset designated for training, as to keep a second anomaly free portion for evaluation preventing the potential distortion of metrics due to the presence and imputation of extreme values.

Subsequently, percentile normalization (Eq. 48) is applied to mitigate extreme values. This normalization process enhances the convergence of optimization algorithms and prevents outliers from disproportionately influencing the data, while maintaining meaningful variation in the regular data.

$$x_i^{norm} = \frac{x_i - p_{0.01}}{p_{99.99} - p_{0.01}} \tag{48}$$

The Industrial Park dataset exhibits missing data, which we address by filling entire days without observations with values from the preceding or succeeding week if is has no missing values. In instances of isolated missing observations, we employ the K-nearest neighbor imputation algorithm to identify days with analogous overall profiles and substitute the missing values with records from the day with the most similar load profile.

Moreover, for the Predis-MHI data, we incorporate extra steps to eliminate periods affected by sensor malfunctions and other exceptional periods characterized by significantly deviant load profiles, such as holidays or the Covid-19 pandemic period when power consumption was negligible.

The AD model is then run on the contaminated portion of the time series (for labeled datasets, we report the anomaly detection performance in Section 3.4.1).

To distinguish between anomaly-free days and those with anomalies, we aggregate patch anomaly scores at the daily level. Specifically, we calculate a day's anomaly score as the highest score within its data, as it was experimentally found that training the AD and AI models on day-long observation to be the most effective.

Anomaly-free day observations are collected and used to train and test the anomaly imputation model by masking random sequences of data and training the model to predict plausible values for them. Once trained, this model is run on days with detected anomalies, i.e. anomalies are masked, and the model is tasked on predicting what the missing values would be under regular conditions.

To evaluate the forecasting accuracy, we train each model on both the raw contaminated data and the same dataset after detecting and imputing anomalies. We assess the performance resulting from each training setting by testing the models on the anomaly-free test data. Moreover, simulating a real-world scenario, we envisioned the model being used daily with data collected from previous days. So, accordingly, we computed performance metrics (see Section 3.3.3) on each day and report the average metric across the entire test period in Section 3.4.3.

Implementation Details

In our experiments, we introduce anomalies into the first 70% of available records, ensuring an equal distribution across the four anomaly types, while the remaining data is kept anomaly-free and reserved to evaluate the forecasting models. In the anomaly detection phase, we augment the time series with additional features before passing them to the anomaly detection phase, Specifically, we incorporate batched standard deviation and generate a new dimension with an exponential smoothed representation to emphasize the contrast between anomalies and their smoothed counterparts, potentially improving anomaly detection performance. Moreover, we employ a frozen ReseNet backbone for extracting patch features at the 1^{st} hierarchy level. Both the anomaly detection and anomaly imputation models are fed day-long data. For training the anomaly imputation model, we randomly omit patches of size 8 (corresponding to the look-up region of the first layer of ResNet.). The embedding dimension is set to 128 and the model is trained using the Adam optimizer with a learning rate of 1e-3.

For forecasting, we employ the previous week data to forecast the next day and leverage the preceding two weeks' data to forecast the upcoming week. The Seq2seq model has one GRU layer with a hidden size of 128 and an FC layer of 16 nodes. The SCINet model is built by stacking two SCINet-blocks of 3 levels. We incorporate a dropout percentage of 25

These hyperparameters have been fine-tuned through iterative experimentation and adjustments on the validation data.

All models are trained using the Mean Squared Error (MSE) loss function and the Adam optimizer for 500 epochs. To prevent overfitting, we implement an early stopping mechanism with patience of 20 epochs [1].

[1]Code for this implementation is made open-source and is accessible at Github.com/MaherDissem/Unsupervised-Anomaly-Detection-in-Noisy-Time-Series-Data-for-Enhancing-Load-Forecasting.

3.3.3 Evaluation

Comparative Baselines

In addition to evaluating the forecasting accuracy gain achieved through our anomaly detection and imputation pipeline, we compare the performance of each individual component against state-of-the-art methods on our custom dataset to provide a comprehensive assessment.

Specifically, we benchmark our anomaly detection technique against DGHL [24], a deep generative model for time series anomaly detection based on top-down Convolutional Networks. DGHL is trained using the Alternating Back-Propagation algorithm combined with short-run MCMC, and has demonstrated state-of-the-art performance in recent comparisons with other advanced anomaly detection models.

For anomaly imputation, we first compare our model against mean imputation, which correspond to filling a missing patch of data with the mean of the time series. To account for temporal trends and changes in the mean, we specifically use the mean from the day on which the anomaly occurs. Although this method is commonly used for single missing data points, it is limited when dealing with multi-timestamp anomalies or anomaly patches, as in our case, and is thus provided for reference only. For more accurate imputation, we also employ the KNN algorithm [17] as a baseline. When imputing a missing sequence (e.g., a 4-hour gap within a day), KNN identifies complete sequences in the training data that are most similar to the surrounding non-missing segments based on Euclidean distance. It then selects the k most similar subsequences and fills each missing timestamp with the average of these subsequences.

Evaluation Metrics

To achieve a comprehensive understanding of the achieved performance in the anomaly detection task, we propose the use of the following key metrics:

- Precision: It measures the proportion of correctly predicted anomalies among all instances flagged as anomalies, indicating the model's ability to avoid false alarms.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{49}$$

In this equation TP are True Positive predictions and FP are False Positives.

- Recall: It quantifies the proportion of actual anomalies that are correctly identified by the model, representing the model's ability to capture all anomalies (i.e. avoiding false negatives).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{50}$$

In this equation FN are False Negative predictions.

- F1 Score: This score harmonizes precision and recall into a single metric, offering a balanced assessment of model performance by accounting for both false positives and false negatives.

$$F1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{51}$$

- AUROC (Area Under the Receiver Operating Characteristic curve): This metric assesses the model's ability to discriminate between normal and anomalous instances regardless of the anomaly score threshold used for classification by measuring the probability that the score of a positive example is higher negative sample's. It ranges from 0 to 1, where a higher AUROC value indicates better performance while an AUROC of 0.5 suggests a model with no discrimination ability (equivalent to random guessing).

Similarly, we adopt the following metrics to evaluate the imputation and forecasting ability of the models where y_i is the actual value at time step i (ground truth) whereas $\hat{y}_i$ is the forecast value and $\bar{y}_i$ is the mean value of the actual observations.

- MAE (Mean Absolute Error): Measures the average across all observations of the absolute errors between predicted and actual values.

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{52}$$

- MSE (Mean Squared Error): Computes the average of the squared differences between predicted and actual values. This metric is more sensitive to large errors than MAE. It is particularly relevant as it is used for the optimization of the ML models. However, it is not as intuitive to interpret due to the values being in a different unit than original values (e.g. KW^2).

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{53}$$

- RMSE (Root Mean Squared Error): Square root of the MSE, providing a measure of the prediction errors in the original unit and relatively sensitive to outliers.

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{54}$$

- sMAPE (symmetric Mean Absolute Percentage Error): The idea behind this metric is to calculate errors as a percentage of the actual values (lower is better), therefore providing a scale independent metric. However, as this can introduce asymmetry due to the impacts of overestimation and underestimation errors being different, it is normalized by the sum of predicted and actual values. This extra step also enhances robustness to very small ground truth values.

$$\text{sMAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2} \times 100\% \tag{55}$$

- R^2 (Coefficient of Determination): Provides a measure of how well the predicted values from the model fit the actual data, relative to a simple horizontal line (naive prediction). This is obtained through the proportion of the variance in the dependent variable that is predictable from the independent variable. A value of 0 indicates that the model's predictions are no better than simply using the mean of the observed data (horizontal line), and 1 indicates a perfect fit where the model perfectly predicts all variations in the data. Negative values indicate that the model is performing worse than a horizontal line.

$$\text{R}^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2} \tag{56}$$

3.4 Results

In this section, we present the results obtained through the methodology outlined in the previous section, as well as additional experiments conducted. We begin by evaluating the anomaly detection and imputation framework independently. Then, we assess the performance improvements achieved using this approach for both the next day and next week forecasting tasks.

3.4.1 Anomaly Detection

We report in Table 3.1 the patch-level anomaly detection metrics obtained for our experiments, conducted under a total contamination rate of 10% where anomalies are spread across 40% of days.

For the AEMO dataset, our model achieves an average F1 score of 92% and an average AUC of 96% across all five time series, demonstrating its strong ability to distinguish between anomalous and regular patterns. This represents a significant improvement over the DGHL baseline, which only achieved an average F1 score of 80% and an AUC of 79%. Similarly, on the Park dataset, our method yields an average F1 score of 90% and an AUC of 96%, outperforming the baseline's scores of 73% for F1 and 72% for AUC.

This performance difference can be attributed to several factors. DGHL lacks a specific mechanism for handling anomalies present in the training data, and was originally designed to leverage spatial correlations in raw multivariate data. However, this approach is less effective for load data, where anomalies typically occur only in the load sensor and not across related sensors. In contrast, our method focuses on feature engineering, extracting a large set of representations from a single time series (e.g. 512 representations using a ResNet backbone in our experiments).

Overall, we obtain good anomaly detection results, with an average AUC score of 96% and F1 score of 92% across all load datasets. Moreover, while the F1 score provide a balance of precision and recall, we are not too concerned with the precision metric, which measures the proportion of flagged anomalies that are actual anomalies, because false positives have low impact on the forecasting models' training since they are replaced by new accurate values. In contrast, we place more emphasis on the recall metric of the anomaly detection (i.e. existing anomalies that are actually detected as such) for whom we obtain an average value of 0.95% by intentionally setting a low threshold on anomaly scores.

Furthermore, to assess the method's effectiveness across various anomaly types beyond just load anomalies, we conduct evaluations on the well-known Yahoo Webscope S5 A2 dataset [128] which contain synthetic labeled anomalies of various kinds, and report competitive results in Table 3.1. Our method achieved an F1 score of 82%, significantly outperforming the DGHL baseline, which only reached 17%. For AUC, we obtained a score of 98%, compared to DGHL's 94%. It's important to note that this dataset has a very low anomaly ratio, with only 0.3% anomalies, indicating a significant class imbalance and AUC is highly sensitive to class imbalance, as it tends to favor the larger class and can yield unreliable scores in such cases. This is because this metric measures the model's ability to rank positive instances higher than the most common negative ones, independent of the classification threshold. In contrast, metrics like precision, recall, and F1 score are more directly influenced by the distribution of positive and negative samples, especially when positive instances are rare. This explains the disparity between the high AUC and the relatively lower F1 score.

However, considering the recognized limitations of such AD benchmark datasets [124], we later shift our focus to directly measuring the impact of anomaly removal by assessing the boost achieved in forecasting accuracy after cleaning the data through our proposed system.

Moreover, it is important to note that, while we use a patch-level evaluation for our AD method, since we are imputing patches of anomalies, for DGHL, we adopt the evaluation algorithm proposed by its authors. This algorithm marks an entire sequence (possibly multiple patches) as anomalous if even a single anomalous data point is detected. While this approach is suitable for detecting anomalies in real-time systems, where immediate action is required to take a system back to its regular behavior, it is less appropriate for imputing anomalies in historical data, where the exact start and end of an anomaly event is unknown. This suggests that DGHL may not be ideal for a combined anomaly detection and imputation approach, as its performance may not translate effectively to our context.

Finally, as the Predis-MHI dataset contains genuine unlabeled anomalies, it is impossible to evaluate and report anomaly detection metrics for this dataset. In fact, the unlabeled nature of this dataset and lack of prior knowledge about it makes our task more challenging. While we are able for

Table 3.1: Evaluation of the Anomaly Detection model on the synthetic datasets (10% contamination rate) and comparison with other techniques

Dataset	DGHL				Ours			
	Precision	Recall	F1	AUC	Precision	Recall	F1	AUC
AEMO/NSW	0.77	0.89	0.83	0.77	0.87	0.97	0.92	0.98
AEMO/QLD	0.79	0.78	0.78	0.80	0.88	0.98	0.92	0.94
AEMO/SA	0.74	0.87	0.80	0.78	0.88	0.96	0.91	0.95
AEMO/TAS	0.76	0.88	0.81	0.81	0.87	0.96	0.92	0.93
AEMO/VIC	0.73	0.90	0.80	0.79	0.87	0.97	0.92	0.98
Park/Commercial	0.74	0.84	0.79	0.72	0.89	0.96	0.93	0.96
Park/Office	0.72	0.82	0.77	0.71	0.85	0.97	0.90	0.97
Park/Public	0.79	0.51	0.62	0.72	0.84	0.96	0.90	0.96
Park/Residential	0.70	0.81	0.75	0.73	0.84	0.96	0.90	0.97
YahooA2 benchmark	0.13	0.26	0.17	0.94	0.74	0.94	0.82	0.98

example to systemically try different thresholds on the anomaly scores of the labeled validation data to find the optimal one before processing the data, we are not able to fine-tune such parameters of the anomaly detection and imputation pipeline for optimal accuracy before moving to forecasting. Hence, we acknowledge the importance of human expertise and feedback in further optimizing the accuracy of our system during the fine-tuning process before deploying the model in a real-life scenario.

3.4.2 Anomaly Imputation

We report in Table 3.2 the performance metrics achieved by the imputation models on the test data across all datasets, measuring the discrepancy between the manually masked patches from the test data and the corresponding original values. To assess this performance, we depend on the accuracy of the anomaly detection model in identifying days with anomalies, ensuring they are excluded from training and testing the AI model. Specifically, the reported imputation performance is assessed by training and evaluating the model using samples identified as anomaly-free by artificially omitting values.

The mean imputation method yields sMAPE scores ranging from 22% to 54% for the AEMO dataset, from 31% to 94% for the Park dataset, and a score of 125% for the Predis-MHI dataset. In comparison, our RDAE model demonstrates significantly higher accuracy, with sMAPE scores ranging from 6% to 12% for AEMO, from 8% to 18% for Park, and achieving a 101% score for

Predis-MHI. Additionally, RDAE outperforms the KNN baseline, achieving sMAPE scores that are, on average, 4.6% lower for AEMO and 2% lower for Park. However, for the Predis-MHI dataset, the KNN imputer is more accurate. This is because the patterns KNN uses for imputation align more closely with the noisy characteristics of the dataset since they are retrieved from the training data, while the Autoencoder generates a smoother curve, resulting in high sMAPE scores for this data. Despite this, other metrics suggest the Autoencoder provides better overall accuracy. In this benchmark we do not report the R^2 score because it can be unreliable for short sequences as it may fit to the noise in the data rather than the underlying patterns.

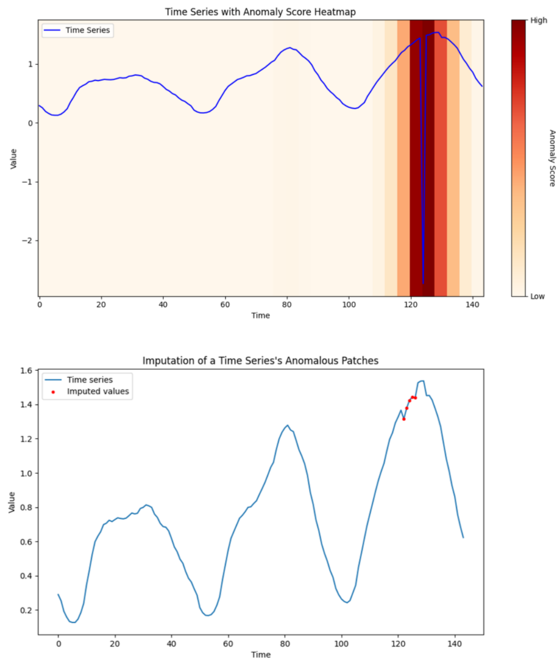
In Fig. 3.5, we compare the time series sequence generated by all methods with the original values that are masked during imputation, on two samples from the AEMO and the Predis-MHI dataset. This reveals that the RDAE sequence follows the overall evolution of the time series more closely than the other methods.

Overall, our model is fairly accurate and since anomalies are spread across a significant portion of the available data, we showcase that the imputation model is able to achieve high accuracy despite only being trained on 60% of total train data (days flagged as anomaly-free by the AD model).

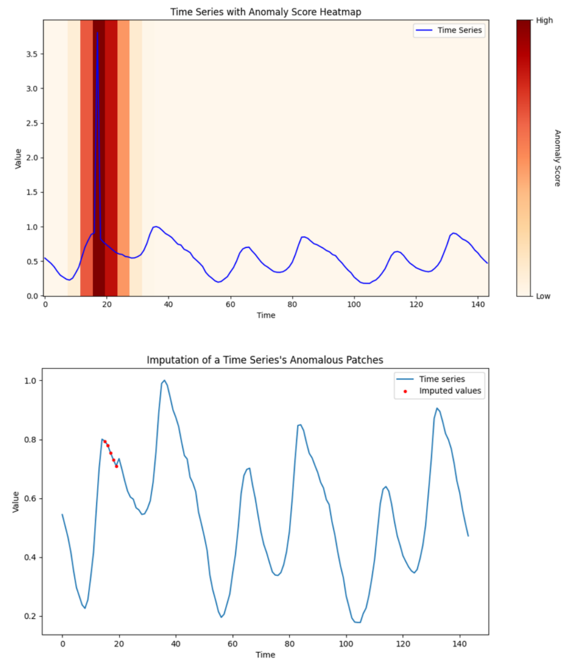
Finally, we further illustrate in Fig. 3.4 how anomalies are detected through an individual patch's anomaly score and consequently replaced by new plausible values, for multiple examples of different load anomalies. We showcase an anomaly of type 3 in examples 3.4(a) and 3.4(c), and an anomaly of type 4 in example 3.4(b). The AD method effectively identifies the patches containing load spikes, assigning the highest anomaly scores to them. Neighboring patches also receive relatively high scores due to the method's ability to create locally aware features. Anomalous patches are then masked and processed by the imputation model to generate plausible values. In example 3.4(d), we showcase an anomaly of type 1, possibly concerning an issue in the recording of load measurements by the energy sensor. Although this type of anomaly is relatively easy to detect manually, it highlights how our combined pipeline can replace such disruptive patterns with plausible values without affecting the rest of the sequence. This approach surpasses simple data cleaning methods such as smoothing data to reduce the impact of noise and spikes [93], which would affect the whole sequence in case of such anomalies.

Table 3.2: Evaluation of the Anomaly Imputation model on its test data and comparison with other techniques

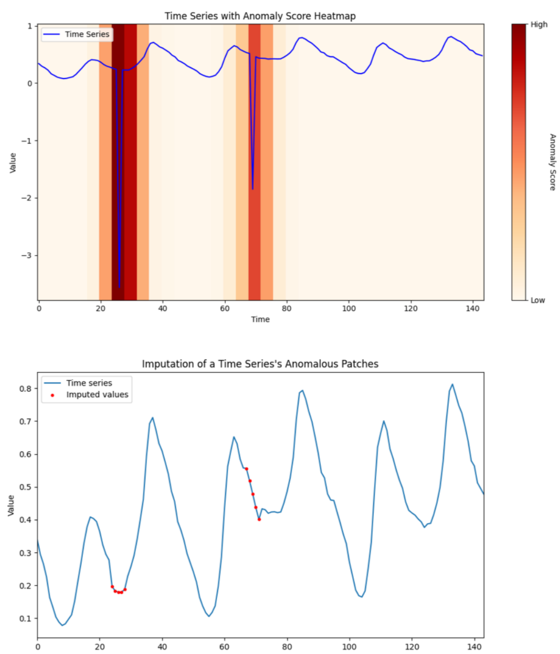
Dataset	Mean Imputation				KNN Imputation				Ours			
	MAE	MSE	RMSE	sMAPE	MAE	MSE	RMSE	sMAPE	MAE	MSE	RMSE	sMAPE
AEMO/NSW	924.02	135.74m	979.32	47.73	222.32	130.10k	237.73	13.68	139.89	32.94k	152.33	12.02
AEMO/QLD	621.67	541.05k	667.06	53.84	113.24	27.66k	123.48	12.90	78.14	11.02k	85.51	7.53
AEMO/SA	135.82	29.19k	144.88	22.55	63.77	6.35k	70.24	10.78	42.21	2.97k	45.59	6.02
AEMO/TAS	87.86	12.46k	95.29	28.08	36.20	2.28k	40.02	12.21	21.35	754.24	23.78	7.00
AEMO/VIC	597.25	513.32k	632.19	47.28	148.23	43.09k	160.07	14.50	79.04	11.68k	84.84	8.41
Park/Commercial	133.07	31.15k	143.09	31.58	34.29	2.12k	40.16	8.87	30.94	2.35k	35.13	8.66
Park/Office	299.47	137.09k	304.81	93.71	28.39	2.00k	35.63	19.53	24.29	1.38k	28.46	17.84
Park/Public	69.56	10.54k	75.02	50.72	19.79	888.26k	23.74	21.30	17.12	601.83	19.62	17.12
Park/Residential	48.51	3.78k	54.00	34.52	13.94	392.44	16.90	10.82	12.52	322.14	14.73	8.78
Predis-MHI	1.90	7.31	2.18	124.92	1.05	2.50	1.32	82.13	0.88	1.69	1.03	100.95



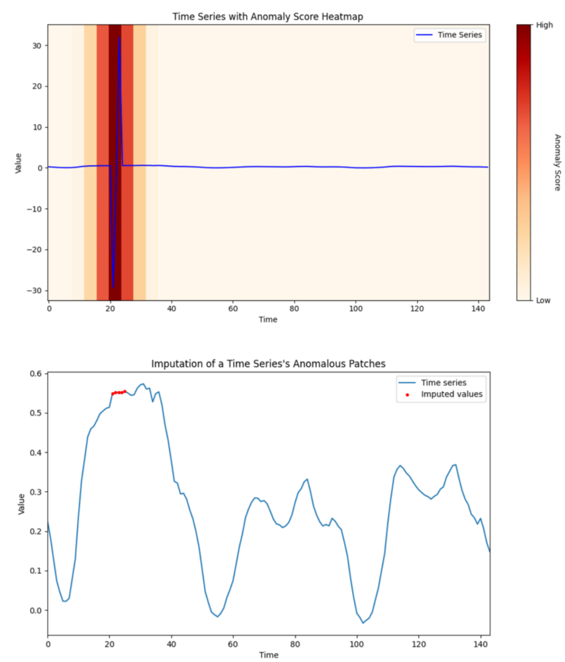
(a) Example 1



(b) Example 2

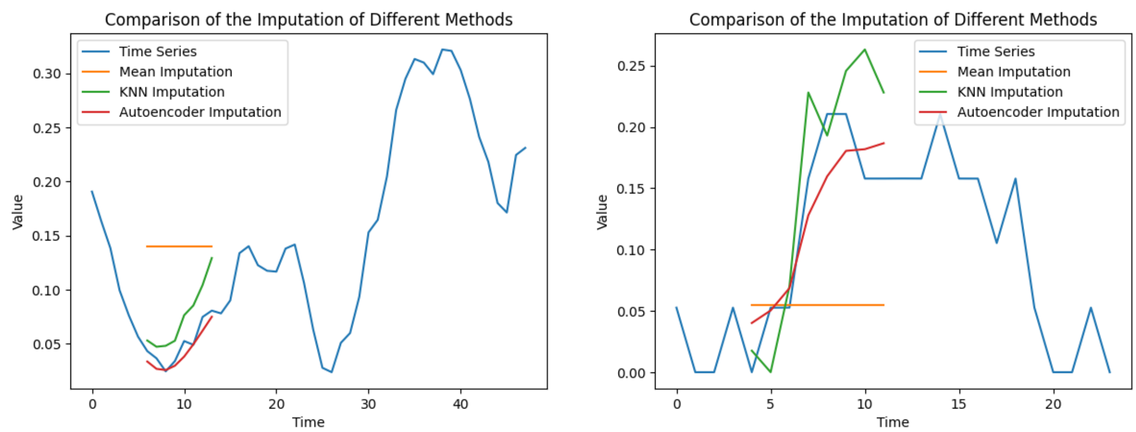


(c) Example 3



(d) Example 4

Figure 3.4: Examples of various anomalies detected in the AEMO dataset: anomaly score heatmap (top) and corresponding imputed patches (bottom)



(a) Example from the AEMO dataset for the Victoria state (VIC)

(b) Example from the Predis-MHI dataset

Figure 3.5: Visual comparison of the imputation generated by different methods for two masked time series samples

3.4.3 Load Forecasting

Next-day forecasting

In Table 3.3, we present the performance improvements attained through our anomaly detection and imputation technique for the Seq2seq model in next-day forecasting scenarios across various datasets and similarly, Table 3.4 showcases the corresponding improvements achieved with the SCINet model. Additionally, Fig. 3.6 illustrates several comparative examples of forecasts generated by models trained on data containing anomalies versus those trained on data post-anomaly detection and imputation. These comparisons illustrate the efficacy of models trained on data containing anomalies versus those trained on the same data post-anomaly detection and imputation, across diverse datasets, models, and forecast horizons.

For the Seq2seq model, notable improvements are observed particularly in the case of the AEMO dataset, where the R^2 score demonstrates a substantial growth from initially poor, unacceptable values (ranging from -0.32 to 0.24) to considerably good ones, averaging 0.6. It's important to note that the MAE, MSE, and RMSE values appear large for the AEMO dataset due to the time series original values being very large as they represent load data aggregated at the state level.

The performance enhancement is comparatively smaller for the Park dataset. This may be attributed to this second dataset being overall more noisy and inherently having irregularities as part of the normal patterns. In response, the model is compelled to refine its learning mechanisms, learning more robust representations of the data that help discern meaningful trends from noise and therefore leading to anomalies having a lower impact on its performance accuracy. Nevertheless, the accuracy boost is still significant with an average decrease of 9% in error (measured in terms of sMAPE) across various building types, with an even more remarkable reduction of 15.14% specifically for public buildings. Moreover, in Fig. 3.6, the first plot illustrates the impact of anomaly presence in the training data on the next day load forecast for a commercial building. Here, we visually contrast the forecasts generated by models trained on cleaned data against those trained on anomaly-contaminated data. The model trained on cleaned data follows the load patterns more closely, capturing fluctuations more accurately, whereas the model trained on anomaly-contaminated data only manages to predict the overall shape of the load patterns without accounting for potential fluctuations.

In contrast, the improvement observed for the SCINet model is slightly less pronounced, given its inherent robustness to noise and anomalies. Nevertheless, the enhancements remain noteworthy, with an average increase of 0.19 points in the R^2 score observed across all Australian states in the AEMO dataset. Additionally, for the Park dataset, a notable improvement of 0.2 is achieved specifically for the Public data subset.

While the final precision of both models is comparable, the Seq2seq model outperforms the SCINet model notably in terms of accuracy when applied to the Predis-MHI dataset. Specifically, training on cleaned data yields significantly superior results for the Seq2seq model, with a 0.09 point higher R^2 score and a mean absolute error that is 0.26 KW lower than for the SCINet model. The main difference between the two models, as suggested in Fig. 3.6, is that the Seq2seq model produces smooth curves that globally follow the load trend, whereas the SCINet model outputs a more erratic sequence that may loosely follow the load values. For this particular dataset, the actual load time series exhibits considerable noise, characterized by frequent fluctuations and small spikes (which are not deemed anomalies due to their high frequency). This noisy nature of the data contributes to noticeably high error metrics, despite both models successfully predicting the overall shape of the next day's load. Nonetheless, after the detection and imputation of anomalies, we observe a significant enhancement in performance, with a 0.14 point higher R^2 score and a 0.39 KW reduction in MAE error for the Seq2seq model. Furthermore, we conduct a qualitative analysis to provide additional insights into the generated forecasts. Notable improvements are observed in certain instances. For instance, we illustrate in Fig. 3.6(b) the electric load prediction for a sequence of 8 days of load data beginning on a Saturday, during which the power consumption is low due to the building being closed. Solely based on load patterns from the previous week, the model trained on cleaned data accurately predicts that the load for the next Saturday will be lower than for weekdays and similar to the pattern at the start of a sequence. In contrast, the model trained on anomaly-infected data fails to identify periods of reduced load during weekends that follow five consecutive days of similar load profile, due to the disruptive influence of anomalies.

Overall, These results indicate that the performance enhancement of the load forecasting is applicable across various building types (residential, commercial, etc.) and remains consistent regardless of the data aggregation rate as it is noticeable for both single-building scenarios, such as the Park dataset, where it can enhance energy efficiency for building owners and energy suppliers, and also in datasets like AEMO, which aggregate load data from numerous buildings at the level of a whole state, offering interesting implications for energy generation, distribution and infrastructure planning as previously mentioned.

Next-week forecasting

In Table 3.5 and 3.6 we respectively present the obtained results for the Seq2seq model and for SCINet model on forecasting the load of a whole week.

For this second task, we encounter an overall decrease in accuracy in comparison to only forecasting one day. This is due to the inherent complexity associated with learning how subtle nuances in long data sequences affect some future patterns but not others. Moreover, this task requires models to retain knowledge over longer sequences, presenting additional challenges such as vanishing gradients or exploding gradients, particularly in the presence of anomalies.

Interestingly, our analysis reveals that the Seq2seq model exhibits poor performance across all metrics when applied to the AEMO dataset. Upon further investigation, we observed that the model tends to converge towards outputting a horizontal line, indicating its inability to effectively learn relevant long-term dependencies and produce accurate forecasts for long forecast horizons within this particular dataset. This issue would be fixed by opting to utilize the SCINet model instead, which demonstrates significantly better forecasting accuracy. On average, the SCINet model achieves a 0.54 higher R^2 score compared to the Seq2seq model, with the sMAPE metric showing a reduction by approximately half when compared with the same model being trained on anomaly infected data. This notable improvement is visually illustrated in Fig. 3.4(d), where only the forecast of the model trained on cleaned data closely follows the actual load curve.

In the case of the Park dataset, both models exhibit impressive final results, with slightly superior performance observed for the SCINet model. However, the SCINet mode does not benefit as much from the anomaly removal process, suggesting that the impact of anomalies on the training process may not always be substantial. This may be attributed to this model's inherent robustness to anomalies, specially that the patterns within this dataset inherently possess higher levels of noise that the model learn to handle during its training.

Finally and in line with the findings for next day's forecasts, the performance of the Seq2seq model on the Predis-MHI dataset, while initially poorer on contaminated data, surpasses that of the SCINet model by a considerable margin after anomaly handling. This is despite the SCINet model performing better on all other time series In fact, detecting and imputing anomalies enhances the Seq2seq model's final performance by a substantial 0.2 points in terms of R^2 score and a reduction of 0.49 MAE.

These results emphasize the importance of recognizing that no single model outperforms others across all datasets, and that a model's robustness to noise and anomalies should not be assumed for any dataset.

Overall, while some cases reveal that the final performance of the models after anomaly processing is still not optimal, highlighting limitations of the forecasting models on certain datasets, the overall picture demonstrates that our anomaly detection and imputation framework significantly enhances the performance of forecasting models across various models and data types.

Table 3.3: Comparison of next day forecasting accuracy across the different datasets for the Seq2seq model trained on raw contaminated data versus one trained on the same data but after detecting and imputing anomalies (10% contamination rate)

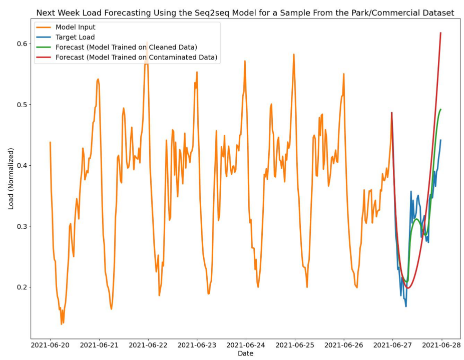
Dataset	Contaminated data					Imputed anomalies				
	MAE	MSE	RMSE	sMAPE	R^2	MAE	MSE	RMSE	sMAPE	R^2
AEMO/NSW	847.23	1325.65k	1143.75	53.44	0.01	451.50	452.94k	670.16	30.04	0.66
AEMO/QLD	663.25	695.02k	832.96	43.63	0.24	263.29	138.52k	372.11	19.75	0.85
AEMO/SA	242.98	110.31k	330.21	38.64	0.11	178.96	63.74k	247.41	30.94	0.51
AEMO/TAS	96.28	14.63k	120.69	50.29	-0.32	52.58	4.56k	67.45	29.68	0.59
AEMO/VIC	582.80	678.34k	820.18	51.38	0.01	406.50	406.79k	630.43	36.99	0.41
Park/Commercial	137.52	36.65k	190.14	38.03	0.76	98.00	19.66k	138.23	27.95	0.87
Park/Office	114.85	28.56k	167.88	58.37	0.71	84.12	17.34k	130.74	47.14	0.82
Park/Public	38.02	2.24k	47.34	76.08	0.17	30.38	1.63k	40.31	66.65	0.40
Park/Residential	47.57	4.36k	65.57	47.86	0.55	35.64	2.92k	53.34	32.71	0.70
Predis-MHI	1.93	11.58	3.37	102.48	0.31	1.54	9.27	3.02	100.25	0.45

Table 3.4: Comparison of next day forecasting accuracy across the different datasets for the SCINet model trained on raw contaminated data versus one trained on the same data but after detecting and imputing anomalies (10% contamination rate)

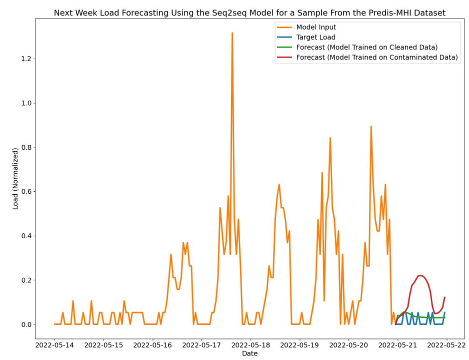
Dataset	Contaminated data					Imputed anomalies				
	MAE	MSE	RMSE	sMAPE	R^2	MAE	MSE	RMSE	sMAPE	R^2
AEMO/NSW	582.79	825.18k	896.93	38.71	0.40	431.76	433.46k	650.16	27.68	0.68
AEMO/QLD	412.11	351.20k	591.43	29.34	0.62	256.89	130.15k	360.27	19.16	0.86
AEMO/SA	193.95	88.32k	291.02	32.82	0.32	166.99	60.17k	240.18	29.26	0.54
AEMO/TAS	67.30	7.29k	85.37	38.62	0.34	57.96	5.39k	73.29	32.76	0.51
AEMO/VIC	456.70	516.44k	714.11	43.04	0.25	372.64	363.54k	594.80	33.13	0.48
Park/Commercial	100.23	23.15k	149.35	26.80	0.85	93.08	19.94k	138.90	25.32	0.87
Park/Office	82.64	20.16k	140.72	38.87	0.79	81.98	20.34k	140.84	42.47	0.79
Park/Public	27.58	1.37	36.99	58.90	0.50	20.47	810.60k	28.43	56.02	0.70
Park/Residential	34.80	3.00k	54.07	30.74	0.69	33.09	2.85k	52.57	29.66	0.71
Predis-MHI	1.74	10.59	3.23	110.16	0.36	1.80	10.69	3.25	112.91	0.36

Table 3.5: Comparison of next week forecasting accuracy across the different datasets for the Seq2seq model trained on raw contaminated data versus one trained on the same data but after detecting and imputing anomalies (10% contamination rate)

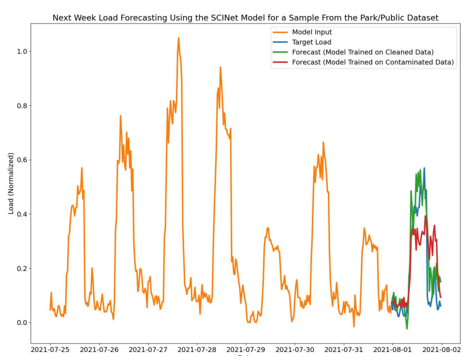
Dataset	Contaminated data					Imputed anomalies				
	MAE	MSE	RMSE	sMAPE	R^2	MAE	MSE	RMSE	sMAPE	R^2
AEMO/NSW	856.28	1273.60k	1126.40	54.48	0.00	815.80	1154.34k	1072.58	51.07	0.09
AEMO/QLD	759.26	928.46k	962.08	49.25	-0.05	696.57	833.80k	911.50	45.36	0.05
AEMO/SA	257.06	133.37k	364.39	40.20	-0.01	249.12	124.99k	352.67	39.28	0.05
AEMO/TAS	88.09	11.74k	108.35	43.58	-0.16	77.01	9.35k	96.68	39.07	0.08
AEMO/VIC	565.30	605.40k	777.59	50.19	0.00	539.83	575.46k	757.82	47.50	0.05
Park/Commercial	162.83	51.52k	226.80	40.76	0.66	147.47	46.90k	215.93	38.56	0.69
Park/Office	144.17	39.40k	198.10	69.50	0.60	103.76	26.20k	161.36	52.47	0.73
Park/Public	40.19	2.59k	50.89	84.23	0.01	29.80	1.44k	37.92	70.53	0.45
Park/Residential	52.86	4.99k	70.38	59.43	0.46	40.73	3.54k	59.04	35.54	0.63
Predis-MHI	1.95	12.96	3.60	114.88	0.18	1.46	9.77	3.12	112.33	0.38



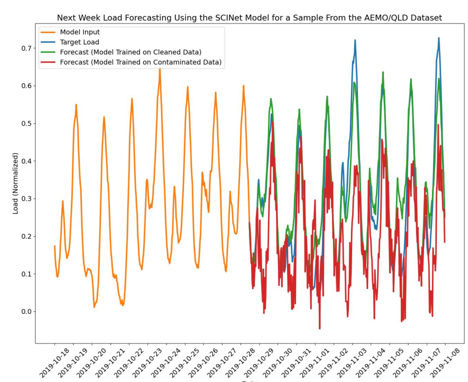
(a) Next day load forecasting sample for the Park/Commercial dataset using the Seq2seq model



(b) Next day load forecasting sample for the Predis-MHI dataset using the Seq2seq model



(c) Next day load forecasting sample for the Park/Public dataset using the SCINet model



(d) Next week load forecasting sample for the AEMO/QLD dataset using the SCINet model

Figure 3.6: Examples from various datasets and forecasting models comparing the forecasting accuracy achieved by training on anomaly-infected data versus training on cleaned data. The labels on the x-axis correspond to midnight of each day.

Table 3.6: Comparison of week day forecasting accuracy across the different datasets for the SCINet model trained on raw contaminated data versus one trained on the same data but after detecting and imputing anomalies (10% contamination rate)

Dataset	Contaminated data					Imputed anomalies				
	MAE	MSE	RMSE	sMAPE	R^2	MAE	MSE	RMSE	sMAPE	R^2
AEMO/NSW	838.79	1427.82k	1191.84	64.17	-0.12	472.51	500.57k	707.18	31.45	0.60
AEMO/QLD	656.11	760.59k	870.48	52.65	0.14	293.36	172.23k	414.23	20.91	0.80
AEMO/SA	208.45	100.66k	315.51	34.98	0.25	206.72	95.32k	307.21	34.54	0.28
AEMO/TAS	100.20	15.72k	125.38	62.28	-0.55	65.45	6.91k	83.09	34.76	0.32
AEMO/VIC	538.66	637.44k	797.86	55.01	-0.06	386.93	387.40k	621.72	35.35	0.36
Park/Commercial	131.05	38.59k	196.00	35.92	0.75	134.38	42.09k	204.70	36.48	0.72
Park/Office	88.31	21.98k	147.57	43.07	0.78	82.25	21.01k	143.88	44.86	0.79
Park/Public	26.77	1285k	35.86	66.48	0.51	19.65	796.58k	28.21	56.53	0.70
Park/Residential	40.56	3.55k	59.02	36.08	0.63	39.59	3.39k	57.78	34.92	0.64
Predis-MHI	1.77	10.89	3.30	117.12	0.31	1.82	10.83	3.29	117.45	0.31

3.5 Discussion and Future Work

In sum, our study demonstrates significant improvements in forecasting accuracy achieved through anomaly detection and imputation across diverse datasets and forecasting horizons. The observed enhancements underscore the broad applicability of our approach, as evidenced by consistent improvements across different building types and data aggregation rates. These findings hold for different forecasting models, and we demonstrate that the optimal model for a given scenario highly depend on the data type and forecasting horizon, as does its anomaly robustness.

The anomaly detection framework, while effective in assigning anomaly scores to samples, faces a notable limitation: the need to set a threshold on these scores to first denoise the memory bank and later, during inference, to distinguish anomalies from regular data points. This process requires prior knowledge of the anomaly contamination rate, which may involve expert input, manual data analysis, or iterative experimentation. Our current strategy favors setting a low threshold, prioritizing recall over precision, given that our imputation method minimizes the effect of false positives. However, future research could explore automating this threshold-setting process.

Additionally, we employ a frozen backbone neural network for the patch feature extraction phase and while it results in competitive performance, future research might explore training a distinct neural network on a time series related task and explore the transferability of its knowledge to the anomaly detection task, in order to achieve more relevant representations in the feature space.

Furthermore, while our experiments focus on univariate time series forecasting, many real-world scenarios capitalize on the interconnectedness of multiple data streams to enhance forecast accuracy. Therefore, a potential avenue for future research could involve exploring anomalies across various sensor types to further evaluate the proposed system's performance in multivariate forecasting scenarios.

3.6 Conclusion

In this study, we introduce an innovative unsupervised memory-based framework designed for anomaly detection within time series data affected by anomalies. We impute identified anomaly sequences with plausible patterns, leveraging an RNN-based recurrent denoising autoencoder trained by randomly corrupting and reconstructing data samples deemed anomaly-free. To quantify the forecasting accuracy gained through of our methodology, we conduct a thorough evaluation by comparing the performance of two substantially distinct models trained both before and after the anomaly detection and imputation processes.

Our approach demonstrates versatility and effectiveness across diverse building types and data aggregation rates, underscoring the importance of managing anomalies in historical data to enhance the reliability of load forecasts, thereby enabling more informed decision-making in energy management and ultimately paving the way for more sustainable and efficient energy utilization and contributing to the realization of smarter and more resilient buildings and electric grids.

Chapter 4

Towards Efficient Federated Load Forecasting: Personalization Mechanisms and Their Impact

4.1 Introduction

Time Series forecasting is generally based on the use of deep learning models that typically require large datasets to be effective, presenting a challenge for new buildings or buildings with new occupants as they often lack sufficient historical data to train their own models, and DL models are known to degrade in performance when trained on small datasets [15]. transfer learning has been proposed as a potential solution, allowing models trained on one dataset to be adapted to a different but related dataset [121, 127, 120]. This approach depends heavily on the similarity between the source and target datasets, and if this similarity is weak, the transferred model may not perform well on the target data [129]. This is particularly problematic for buildings with unique energy consumption patterns.

However, and in addition to data quantity concerns, privacy is a major issue in the building sector, leading to numerous regulatory limitations on data accessibility and sharing [69]. Building energy usage data can reveal sensitive information about the occupants' habits and behaviors, such as daily routines or occupancy patterns [104]. These insights could potentially be exploited, raising privacy concerns that make many building owners reluctant to share their data, whether for TL or for centralized model training.

Federated Learning has emerged as a promising approach to exploit data from multiple sources in a privacy-conscious way [84]. In FL, each client (i.e., building) trains a model locally using its own data, sharing only the model updates with a central server instead of the raw data itself. This method ensures client privacy, as no sensitive data leaves the client's environment, while also allowing clients to benefit indirectly from the aggregated knowledge gained from the model updates of other clients.

In standard FL, a single global model is built for all participating clients by aggregating their model updates. While this approach works well for a client pool of similar data distributions, it struggles in scenarios where client data is heterogeneous, a common situation in the building sector. This "curse of data heterogeneity" arises when the data across clients is not Independent and Identically Distributed (non-IID) as buildings may vary significantly in size, configuration, regional climate, and occupant behavior.

This heterogeneity introduces significant challenges: one type of load profile may become over-represented in the federation, causing the global model to primarily learn its dominant patterns. This can lead to negative learning, where the patterns of the underrepresented clients are overwritten by the majority, degrading their performance. Moreover, the disparity in data distributions can slow or prevent the convergence of the global model, causing it to reach a suboptimal solution that fails to capture the unique patterns of all clients, resulting in suboptimal forecasting performance for distributions that are significantly different from the global one [109]. Without a centralized comprehensive view of all data distributions, it is impossible to analyze the entire pool of clients to account for their diverse data patterns and try to solve this problem with conventional approaches like data rebalancing or data augmentation [109]. This challenge has led to the emergence of a new field within federated learning known as Personalized Federated Learning, which seeks to develop a federated model that leverages knowledge from the entire client pool while emphasizing the unique characteristics of individual clients. Numerous approaches have been proposed, in different contexts tailored to specific types of data, including images, text and time series. However, to the best of our knowledge, no comprehensive study has been conducted to compare these approaches on the specific task of load forecasting, making it crucial to evaluate their performance under consistent conditions.

In this research, we aim to investigate the benefits of personalization in federated load forecasting. Therefore, we select a single forecasting model and compare integrating it to different PFL approaches. We opt for the SCINet model used in the previous chapter and described in section 3.2.3, because in [80] it demonstrates state-of-the-art performance on a benchmark including several other recent models. For FL personalization, we select recent methods that have shown strong performance and represent distinct personalization archetypes. We do not consider in this study the training and communication costs of FL or potential forecasting improvements from further model or data engineering, but focus exclusively on raw forecasting performance to evaluate the benefits of personalized federated learning with minimal forecasting fine-tuning.

The rest of this chapter is organized as follows: in section 4.2 we introduce and explain the federated load forecasting framework and the different FL personalization mechanisms that we evaluate. In Section 4.3, we detail the experimental setup, including the datasets used, the specific methodologies employed in our implementation and the evaluation process of the whole system. Following this, Section 4.4 presents a comprehensive analysis of the experimental findings, highlighting the performance of each personalization technique in optimizing load forecasting accuracy. We interpret the results and discuss their implications for future research and practical applications in smart building energy management, while also pointing the limitations of our study and potential directions for further exploration.

4.2 Analytical Framework Overview

In this section, we present an overview of the federated load forecasting framework, describing how the system leverages decentralized data while maintaining privacy. We then outline the personalization mechanisms applied.

4.2.1 Context and notation

We consider a federated learning setting with a set of clients C of size K, where each client $c_k \in C$ ($k \in \{1, 2, \ldots, K\}$) has access to its own private dataset $D^k = \{(x_i^k, y_i^k)\}_{i=1}^{n_k}$ and public model weights θ_k. Here, $x_i^k \in \mathcal{X}$ and $y_i^k \in \mathcal{Y}$ represents respectively the n_k load sequences fed as input to the model and the corresponding target sequences that it learns to forecast.

For each client c_k, the input sequences x_i^k and target y_i^k are generated using a sliding window mechanism. Given a time series of historical load data, the sliding window process extracts consecutive overlapping input and target sub-sequences of fixed length X and Y, respectively, with a stride parameter δ. Specifically, let $(s_t)_{t=1}^{T}$ be the raw historical load data for a client. The input sequence $x_i^k = (s_{i\delta}, s_{i\delta+1}, \ldots, s_{i\delta+X-1})$ represents the load data over X data points, and $y_i^k = (s_{i\delta+X}, \ldots, s_{i\delta+X+Y-1})$ represents the subsequent target load values that the model must forecast.

The forecasting model for each client c_k is trained by updating its weights θ_k while minimizing a loss function $\mathcal{L}^k(\theta_k)$ that minimizes the difference between the predicted values $(\hat{y})_i^{n_k}$ and actual target values $(y)_i^{n_k}$. More specifically, this loss function for all data samples can be expressed as follows:

$$\mathcal{L}^k(\theta_k) = \sum_{i=1}^{n_k} l(x_i^k, y_i^k; \theta_k) = \sum_{i=1}^{n_k} \frac{1}{Y} \sum_{j=1}^{Y} \|f_\theta(x)_j - y_j\|^2 \tag{57}$$

where f denotes the model and θ represents the weights of all neural networks within the model.

We assume that the data across clients is non-IID, which means each dataset D^k follows a client-specific data distribution P^k. This reflects the inherent variability in the load patterns of different buildings or clients, influenced by factors such as geographical location, building type, and user behavior. Each client owns their data locally, and the privacy constraints of federated learning dictate that this data cannot be shared with the central server or other clients.

The goal of federated learning is for these clients to collaboratively train a global forecasting model hosted on a central server, with model weights θ_g, without sharing any data from $(D^k)_{k=1}^{K}$ with other clients or with the central server. Instead, clients train local models θ_k on their private datasets and share only their model updates with the server. The server aggregates these updates to refine the global model, so that it becomes capable of accurately forecasting energy loads across all clients in the network.

4.2.2 Federated Load Forecasting

In the federated learning context, we use the Federated Averaging (FedAvg) [84] algorithm to update the global model. This algorithm enables clients to collaboratively train the global model while preserving data privacy by transmitting model updates rather than raw data.

At each communication round t, the central server samples a subset of available clients $C^{(t)}$ from the client network and sends them the current global model weights $\theta_g^{(t)}$. Each client c_k then updates its local model $\theta_k^{(t)}$ by minimizing a local loss function $\mathcal{L}^k(\theta_k^{(t)})$, that captures the difference between true load sequences and the sequences predicted using the local model $\theta_k^{(t)}$.

After the local training, each client k computes its model update and sends it to the server. The server aggregates the updates from the clients to compute the new global model weights $\theta_g^{(t+1)}$. The aggregation step in FedAvg is a weighted average of the client models, where the weight is proportional to the size of the local dataset n_k:

$$\theta_g^{(t+1)} = \frac{1}{N} \sum_{k=1}^{K} n_k \theta_k^{(t+1)} \tag{58}$$

where $N = \sum_{k=1}^{K} n_k$ is the total number of samples across all clients.

This process is repeated for multiple communication rounds until the global model converges. The FedAvg algorithm, summarized in 3, provides a simple yet effective way to aggregate decentralized learning processes across multiple clients.

4.2.3 Personalization Approaches for Federated Learning

FedALA: Personalized Federated Learning through Model Initialization

[136] highlights a crucial limitation of FedAvg: when the server aggregates the weights from all client models into a single global model, this new global model is then sent to all clients, overwriting their local models. This process erases the knowledge each local model has learned from its unique data during the previous round, as all clients must restart the training process from a new common initial set of parameters that what might not generalize across all clients in a non-IID data scenario.

Algorithm 3 FedAvg Algorithm

Input: Client population $\mathcal{C}$, number of communication rounds T

Output: Final global model weights $\theta_g^{(T)}$

```
Initialize global model weights θ_g^(0)
for each round t = 0, 1, ..., T do
    Server samples a subset of clients C^(t) from the total client population 𝒞
    Server sends global model weights θ_g^(t) to clients in C^(t)
    for each client k ∈ C^(t) do
        Client k updates local model θ_k^(t) by minimizing local loss ℒ^k(θ_k^(t))
        Client k sends updated weights θ_k^(t+1) to the server
    end for
    Server aggregates local updates:
    θ_g^(t+1) ← (1/N) Σ_{k=1}^{K} n_k θ_k^(t+1)
end for
```

This algorithm suggests an alternative: instead of fully overwriting local models, an adaptive model initialization mechanism should be employed. This mechanism adaptively combines the global model's weights with the local model's previous weights, preserving knowledge learned in earlier rounds while incorporating the new information.

The initialization process for the local model c_k at the start of training is governed by the following equation:

$$\theta_k^{(t)} \leftarrow \theta_k^{(t-1)} + (\theta_g^{(t-1)} - \theta_k^{(t-1)}) \odot W_k^{(t-1)} \tag{59}$$

where $\theta_g^{(t-1)}$ represents the global model weights received from the server, $\odot$ denotes the Hadamard product, and $W_k^{(t-1)}$ is a learnable matrix that assigns aggregation weights to each parameter in $\theta_g^{(t-1)}$ and $\theta_k^{(t-1)}$.

Each client learns its own $W_k^{(t-1)}$ matrix by running an optimization process at the start of each training round. This is done by minimizing the forecasting loss through a gradient descent step, adjusting W_k towards the optimal combination of global and local parameters. This process is described by the equation:

$$W_k^{(t)} \leftarrow W_k^{(t)} - \frac{\eta}{|D^k|} \sum_{(x,y)\in D^k} \nabla_{W_k^{(t)}} l(x, y; \theta_k^{(t)}, \theta_g^{(t)}) \tag{60}$$

where η is the learning rate, and (x, y) represents data samples from the client's dataset D^k.

Other approaches have been proposed to optimize this process computationally, such as only aggregating the weights of higher layers of the neural network, which are more task-specific, or computing gradients on a subsample of data. In this work, however, we prioritize maximizing performance to identify the optimal personalization strategy for personalized federated learning, so we do not incorporate additional optimizations at this stage.

Fedora: Personalized Federated Learning through Adaptive Parameter Propagation

A novel approach to achieving personalization in federated learning is introduced in [123]. It adapts principles from prior work in transfer learning [19] to frame this challenge as a privacy-preserving transfer learning problem. In this approach, each client selectively receives knowledge from others, and simultaneously determines whether and how to transfer that knowledge based on data distribution similarities. Specifically, a client will only consider the weights of another client if their data distributions are similar, as this indicates that their model parameters are likely to be aligned as well. This safeguards the generalization performance, as transferring knowledge from a client with dissimilar data could degrade the receiving client's performance.

In this framework, the server retains auxiliary copies of each client's model weights, denoted as $\hat{\theta}_k$. In other terms, each client c_k shares its weights θ_k with the server, and in return, receives updates weights $\hat{\theta}_k$ that take into account the knowledge of other clients.

Hence, the server is responsible for updating the auxiliary parameters θ_k $(k = 1, \ldots, K)$ by aggregating the weights across clients. For each client c_k, the weights are updated based on the weights of other clients $(\hat{\theta}_j)_{j \neq k}$. However, instead of using a simple weighted average (like FedAvg), the server implements a more sophisticated weighted aggregation approach, incorporating a weight parameter that quantifies the similarity of each client's data distribution to that of the client c_k.

To estimate data similarity without exchanging actual data, the idea is to exploit the training-induced subspaces for each client. To do so, we perform truncated singular value decomposition (SVD) on the training data at the client-side to generate a subspace representation for each client c_k. These subspace representations are sent to the server, where the similarity between a pair of clients,

k and k', is measured as $W_{kk'}$, based on the principal angles between these representation [1].

The update of each auxiliary model at federation round t is defined in the following equation:

$$\hat{\theta}_k^{(t)} = \frac{\alpha}{(1+\alpha)D_{kk}} \sum_{k'=1}^{K} w_{kk'}\theta_{k'}^{(t-1)} + \frac{1}{1+\alpha}\theta_k \tag{61}$$

where α is a hyperparameter that defines the learning rate of the auxiliary parameters and $D_{kk} = \sum_{k'=1}^{K} W_{kk'}$.

From the client's perspective, the process consists of receiving the frozen auxiliary parameters $\hat{\theta}_k$ from the server and updating its local model weights θ_k. Specifically, client c_k updates θ_k to minimize the loss on its local data and evaluates $\hat{\theta}_k$ on a validation set. If $\hat{\theta}_k$ yields a lower validation loss than the locally trained model θ_k, the client incorporates $\hat{\theta}_k$ to guide its local weights towards $\hat{\theta}_k$. Otherwise, it continues regular local training without considering the auxiliary parameters. This is formalized in the following minimization objective:

$$\min_{\theta_k} \frac{1}{|D^k|} \sum_{(x,y)\in D^k} l(x, y; \theta_k) + \lambda_k \|\theta_k - \hat{\theta}_k\| \tag{62}$$

where $\lambda_k = \max(0, l(D^k_{\text{valid}}; \theta_k) - l(D^k_{\text{valid}}; \hat{\theta}_k))$. This term dictates whether the client should benefit from the auxiliary parameters ($\lambda_k > 0$) and determines the balance between local training and incorporating the federated weights.

The adaptive parameter propagation mechanism resembles standard federated learning, with the primary distinction being that it selectively propagates parameters from one model to the other based on data distribution similarity, instead of enforcing a single global model across all clients. This targeted transfer ensures that only beneficial knowledge is shared, effectively mitigating negative transfer: when a local model's performance is adversely affected by parameters from a client with a dissimilar data distribution.

Model-Mix: Personalized Federated Learning through private-public model mixing

[51] proposes achieving model mixture-based personalization. The idea is to have the client

[1] For further details about this process, refer to [123]

train its own local model on its private data, and at the same time to build a second public model that it uses to share essential knowledge with a central server. This server, in turn, aggregates contributions from the public models of multiple clients to create a global federated model that benefits from diverse data exposure, thus capturing broader data patterns across the entire network of clients. This way, the client benefit from this pooled knowledge, gained by collaborating with other clients that are exposed to different data patterns, while still retaining a client-side model component that is optimized for its specific data distribution and characteristics. This dual-model approach strikes a balance between individual personalization and collective knowledge sharing, making it particularly suited for scenarios with heterogeneous client data.

This is done by inducing an explicit connection in weight space between the optima of the local and the federated models, so they both are able to boost each other's performance. This idea is backed by the recent discovery through extensive studies on the loss landscape of deep networks of the existence of a connected subspace contains low-loss solutions for multiple independent models [48, 39]. Specifically, when two or more deep networks are trained independently, it has been demonstrated that their optima lie within a low-loss subspace, a region in the parameter space where models can interpolate smoothly between their respective solutions without a significant increase in loss.

Mathematically, this subspace can be characterized by defining a weight interpolation between two models [119, 51]. Let θ_1 and θ_2 represent the weights of the two models. The optimization process seeks a path in the parameter space Θ that connects these two weight vectors such that the loss function $\mathcal{L}(\theta)$ remains minimal along the path. Formally, this can be written as:

$$\min_{\theta \in \Theta} \mathcal{L}(\theta), \quad \text{where} \quad \theta = \alpha\theta_1 + (1-\alpha)\theta_2, \quad \alpha \in [0,1] \tag{63}$$

This suggests that the models' weights can be linearly interpolated through a convex combination controlled by α, allowing for a smooth transition between the local and global solutions while remaining within the low-loss region of the loss landscape. This approach fosters an efficient exchange of information between local and global models, enhancing both the personalization for each client and the overall generalization across the federated learning system

In this work, we leverage this approach by allowing each client c_k to maintain two separate forecasting models of identical architecture: a local model with weights $\theta_{k,l}$ and a public model with weights $\theta_{k,f}$ for participating in the federated learning framework. The FL process begins with the server initializing the global model $\theta_g^{(0)}$ and transmitting it to the participating clients. Then, in each round t, each client k then sets the federated model weights $\theta_{k,f}$ to the received global weights $\theta_g^{(t)}$ and loads its local model $\theta_{k,l}$ that was pretrained during previous rounds. During the client's forward propagation step, we introduce a stochastic model mixture strategy where, for each mini-batch $B_i^k \in D^k$, a mixing parameter $\alpha \sim \mathrm{Unif}(0,1)$ is sampled from a uniform distribution. This generates a mixed model, defined as:

$$\theta_k(\alpha) = \alpha\theta_{k,l} + (1-\alpha)\theta_{k,f}. \tag{64}$$

Randomly sampling a different α allows the model to explore different convex combinations of the local and federated model, leading to a broader search in the parameter space and allowing the model to identify better solutions that generalize well across both the client-specific and global objectives and prevents it from overly relying on the characteristics of either model alone.

This mixed model θ_k is then used to forecast the target time sequence. A first forecast specific-loss function $l(x, y; \theta_k(\alpha))$ is computed to quantify the discrepancy between the time sequence $\hat{y}$ generated by the mixed weights as a forecast for x, and the actual value y. And to ensure the models remain aligned with both local and global objectives, we introduce a regularization mechanism in the form of two more loss functions. Hence, the total loss function for backpropagation is defined as:

$$\begin{aligned}\mathcal{L} = {} & \frac{1}{|B_i^k|}\sum_{(x,y)\in B_i^k} l(x, y; \theta_k(\alpha)) \\ & + \mu\|\theta_{k,f} - \theta_g\|_2^2 \\ & + \nu\cos^2(\theta_{k,f}, \theta_{k,l})\end{aligned} \tag{65}$$

The second term in the equation $\mu\|\theta_{k,f} - \theta_g\|_2^2$ ensures the federated model $\theta_{k,f}$ remains close to the received global model θ_g, which is kept frozen during local client updates. This term prevents the federated model from diverging too far from the global model, preventing overly aggressive local updates that could introduce noise or overly client-specific knowledge into the federated model, hence maintaining the integrity of the aggregated knowledge from other clients.

The third term $\nu \cos^2(\theta_{k,f}, \theta_{k,l})$ encourages connectivity between the local and federated models by inducing orthogonality between their weight vectors. Besides, minimizing cosine similarity between $\theta_{k,f}$ and $\theta_{k,l}$, prevents the neural networks from learning redundant features and ensures that both models specialize in different aspects of the data. The local model focuses on client-specific knowledge, while the federated model captures more general, client-agnostic patterns. This design reduces the risk of negative knowledge transfer by ensuring that redundant features are not learned by both models, allowing each to contribute uniquely to the forecasting task.

To train this model, we compute gradients for both the local and the federated weights as follows and perform backward propagation with the Stochastic Gradient Descent algorithm.

$$\begin{aligned} \nabla_{\theta_{k,l}} \mathcal{L} = &\frac{\alpha}{|B_i^k|} \sum_{(x,y)\in B_i^k} \nabla_{\theta_k} l(x, y; \theta_k(\alpha)) \\ &+ \nu \frac{\partial}{\partial \theta_{k,l}} \cos^2(\theta_{k,f}, \theta_{k,l}) \end{aligned} \tag{66}$$

$$\begin{aligned} \nabla_{\theta_{k,f}} \mathcal{L} = &\frac{1-\alpha}{|B_i^k|} \sum_{(x,y)\in B_i^k} \nabla_{\theta_k} l(x, y; \theta_k(\alpha)) \\ &+ \mu \frac{\partial}{\partial \theta_{k,f}} \|\theta_{k,f} - \theta_g\|_2^2 \\ &+ \nu \frac{\partial}{\partial \theta_{k,f}} \cos^2(\theta_{k,f}, \theta_{k,l}) \end{aligned} \tag{67}$$

After training on all mini-batches $B_i \in D^k$, each client retains its local model component $\theta_{k,l}$, and transmits the updated federated model $\theta_{k,f}$ to the central server. The server then aggregates the received federated models from all clients into a new global model $\theta_g^{(t+1)}$ for the next round of federation. This aggregation is akin to performing the FedAvg algorithm (Alg. 3), but applied only to the federated component of clients' models.

After the FL process concludes and to optimize the final forecasting performance, clients evaluate the optimal value of the mixing parameter α using their validation data to find the optimal balance in the contributions of the local and federated models for their specific dataset.

This combined approach ensures both robust personalization for each client and effective global generalization, enhancing performance across the entire federated learning framework.

An overall algorithm is provided in 4 to summarize the training procedure.

FedCluster: Personalized Federated Learning through Client Clustering

An intuitive way to address the issue of negative knowledge transfer in federated learning is to group similar clients together, ensuring that clients with vastly different patterns do not interact with each other during model updates.

This clustering approach does not provide individual client-level personalization, since all clients in a cluster share a single model. It can however be combined with traditional FL methods (e.g., FedAvg) to provide more data-tailored global models to clients by isolating some of them and adjusting their participation in the process, leading to cluster-level client personalization. It can also be integrated with personalized FL approaches to assess whether the personalization effectively mitigates negative transfer. If the final performance does not improve, it indicates that clients are not adversely affected by others within their cluster. In such cases, we may conclude that the personalization strategy inherently addresses the issue of the model being negatively influenced by dissimilar client distributions.

To achieve this in a privacy-preserving manner, [52] proposes a novel algorithm for conducting load clustering in a distributed fashion without aggregating user data in a centralized location. The process runs a federated version of the time series K-means clustering algorithm. It starts by identifying cluster centroids. Then, each client is assigned to a cluster based on the distance between their average load profile and the nearest cluster centroid. Once the clusters are formed, standard FL is run independently for each cluster (as described in Algorithm 3), allowing clients within the same cluster to share a single model

More specifically, this process starts with the server initializing Q cluster centroids by sampling a 1D sequence from a uniform distribution for each one and broadcasting them to participating clients. The initial centroids are as follows:

$$\begin{gathered}\Gamma_t = [\mu_{t,1}, \mu_{t,2}, \ldots, \mu_{t,Q}] \\ \mu_{t=0,i} \sim \text{Unif}(0,1)^{X+Y} \quad \forall i \in \{1, 2, \ldots, Q\}\end{gathered} \tag{68}$$

Algorithm 4 Personalized Federated Learning Framework based on Model Mixing

Input: Client population $\mathcal{C}$, number of communication rounds T

Output: Final global model weights $\theta_g^{(T)}$, final personalized model weights $\theta_{k,l}^{(T)}$ and mixing parameter α_k for each client c_k

Initialize global model weights $\theta_g^{(0)}$ and individual client local models $\theta_k^{(0)}$ for $k \in \{1, 2, \ldots, K\}$
for each round $t = 1, 2, \ldots, T$ **do**
    Server samples a set of participating clients $C^{(t)}$
    Server sends global model weights $\theta_g^{(t)}$ to all clients in $C^{(t)}$
    for each client $c_k \in C^{(t)}$ **do**
        Load local model weights $\theta_{k,l}^{(t)} \leftarrow \theta_{k,l}^{(t-1)}$
        Set federated model weights $\theta_{k,f}^{(t)} \leftarrow \theta_g^{(t)}$
        for each mini-batch $B_i^k \in D^k$ **do**
            Sample mixing parameter $\alpha \sim \text{Unif}(0, 1)$
            Compute mixed model weights following Eq. 64
            Compute predictions $\hat{y} = f_{\theta_k^{(t)}(\alpha)}(x)$ for each input $x \in B_i^k$
            Compute the loss $\mathcal{L}$ following Eq. 65
            Compute gradients $\nabla_{\theta_{k,l}^{(t)}} \mathcal{L}$ and $\nabla_{\theta_{k,f}^{(t)}} \mathcal{L}$ following Eq. 66 and Eq. 67
            Update local and federated model weights:

$$\theta_{k,l}^{(t+1)} \leftarrow \theta_{k,l}^{(t)} - \eta \nabla_{\theta_{k,l}^{(t)}} \mathcal{L}$$

$$\theta_{k,f}^{(t+1)} \leftarrow \theta_{k,f}^{(t)} - \eta \nabla_{\theta_{k,f}^{(t)}} \mathcal{L}$$

        end for
        Client c_k sends updated federated model weights $\theta_{k,f}^{(t)}$ to the server and stores its local weights $\theta_{k,l}^{(t)}$
    end for
    Server aggregates local updates:

$$\theta_g^{(t+1)} \leftarrow \frac{\sum_{c_k \in C^{(t)}} n_k \theta_{k,f}^{(t+1)}}{\sum_{c_k \in C^{(t)}} n_k}$$

end for
Each client c_k searches for the optimal α_k parameter on its validation data

where Q represents a hyperparameter that dictates the maximum number of clusters.

For each round t, each client c_k, computes two key vectors: the first, $J_{t,k}$, captures the average distance between each time series $x_i^k \quad \forall i \in \{1, 2, \ldots, n_k\}$ in its historical training data and the centroid closest to it, calculated as follows:

$$J_{t,k} = \sum_{t=1}^{n_k} \omega_{t,k,i}[P_{k,i} - \mu_{t,q}] \quad \forall q \in \{1, 2, \ldots, Q\} \tag{69}$$

where $P_{k,i} = [x_1^k, \ldots, x_{n_k}^k]$ is a matrix representing the historical time sequences in the training data of client c_k and $\omega_{t,k,i} = \text{arg min}_q \left\|P_{k,i} - \mu_{t,q}\right\|_2, \quad \forall i \in \{1, 2, \ldots, n_k\}$ assigns each time series x_i^k to the closest centroid. The second vector, $V_{t,k}$, counts the number of time sequences assigned to each cluster q.

This algorithm avoids the transmission of raw data to the server by instead, having each client sends these two vectors, $J_{t,k}$ and $V_{t,k}$ to the server, which do not reveal direct information about clients' data, ensuring they are kept private.

Upon receiving the vectors from all clients, the server updates the cluster centroids using the following equation:

$$\Gamma_{t+1} = \Gamma_t + \alpha \frac{\sum_k J_{t,k}}{\sum_k V_{t,k}} \tag{70}$$

Here, α is a learning rate that controls how quickly the centroids move towards their optimal positions.

The centroids are updated iteratively until convergence, which occurs either after a preset number of iterations or when the centroid updates become smaller than a predefined threshold. This process ensures that the centroids progressively minimize the distance between themselves and the time series assigned to them across all clients.

Once the centroids have converged, each client assigns itself to the cluster whose centroid is closest to its average load, calculated as:

$$\hat{\omega}_k = \arg\min_q \left\| \frac{1}{n_k} \sum_{i=1}^{n_k} x_i^k - \mu_q \right\|_2 \tag{71}$$

4.3 Methodology

In this section, we start by introducing the data sources for our experiments. Then, we describe how the different components of our system are implemented. Finally, we outline the evaluation protocol used to assess the performance of the proposed approaches.

4.3.1 Datasets

While there does not exist a specific dataset specially tailored for Federated load forecasting research, within our context, we repurpose existing datasets designed for other tasks, that record building energy consumption for multiple different buildings at once.

Since as of our current knowledge, there is no dedicated dataset specifically designed for federated load forecasting research, we adapt existing datasets that capture energy consumption data from multiple buildings, even though they were originally intended for other tasks, to fit our context.

REFIT Dataset The REFIT dataset [85] is a publicly available collection of electrical load measurements obtained from a longitudinal study conducted in the United Kingdom. This dataset provides detailed meter readings, including whole-house aggregate loads and individual measurements from nine appliances in each of the 20 households, captured while occupants followed their regular daily routines. The data were originally recorded at 8-second intervals over a two-year period. In our experiments, we resample the data to hourly intervals and focus on forecasting the total aggregate load for each household.

IRISE Dataset The IRISE dataset [16] is derived from the European Remodece project, which monitored residential energy consumption across various European countries. We utilized a subset of this data, specifically 23 households in France, recorded between January 1998 and April 1999. The dataset includes hourly aggregated power data for most household appliances, along with corresponding weather information such as temperature and humidity. In our experiments, we aimed to forecast the total aggregate load for each house.

Combined Dataset While each of the two datasets offers a diverse range of clients with distinct load profiles, shaped by varying building characteristics, settings, and user behaviors, we also decide to merge the two datasets into a single one to further stress-test the PFL algorithms' robustness. This integration not only increases the number of clients up to 43, but also ensures that each interacts with a larger and more varied group, increasing both the heterogeneity and complexity of the simulation environment.

4.3.2 Implementation Details

To generate the data, we apply a sliding window approach on the raw historical time series data of each building. The sliding window has a fixed size of 7 days, moving forward by one day at each step. The obtained time windows are collected and min-max normalized. For each window, the first 6 days serve as raw input to the model (i.e. without any additional feature engineering), which forecasts the following 24 hours. Hence, using the defined notation, we have $X = 6 \times 24$, $Y = 24$ and $\delta = 24$. The specific choice of a 6-day input window was made to ensure that each window captures sufficient information on both the weekly load and its potential variability. A shorter window could hinder the model's ability to identify whether the day being forecasted is a weekend or a weekday, which typically exhibits different load patterns, while a longer window could complicate the task by introducing excessive information, making it harder to learn complex correlations through these extended horizons.

The average number of extracted windows across all data is 368 samples, with a standard deviation of 38 samples, representing a notably small dataset size that effectively simulates the study scenario of buildings lacking extensive historical data. These windows are divided into training, validation, and test sets, with 70%, 15%, and 15% of data allocated to each, respectively. We compute the evaluation metrics for each forecast day in the test set independently, then average them over all time windows. This simulates a real-world scenario where, for every day, a new 24-hour forecast is generated based on the preceding data. The evaluation metrics are further explained in the next section.

We train a 2-block SCINet model by minimizing the MSE loss for a maximum of 200 epochs, with early stopping applied after 20 epochs without improvement. The model employs a 25%

random neural dropout rate and an Adam optimizer with a decaying learning rate.

The federated learning simulations are implemented using the Flower framework [21] [2]. We run 30 FL rounds, with an additional 10 rounds of clustering when applicable. The reported results correspond to the metrics achieved by clients on their test set for the round where their model achieved the best performance on the validation set. This approach ensures that a building does not replace its model with one that performs worse on local validation data than its previous model. In each round, all available clients are selected for training.

Hyperparameters for both the forecasting model and PFL algorithms are optimized through iterative testing and tuning on the validation data.

4.3.3 Evaluation Metrics

- RMSE (Root Mean Squared Error): It calculates the square root of the average of the squared differences between these values. The squaring emphasizes larger errors, making RMSE particularly sensitive to outliers and to deviations, while taking the square root returns the error to the original unit of measurement (e.g., if the target is in kilowatts, the RMSE will also be in kilowatts). This metric provides an overall sense of how close the predictions are to the actual values, with lower values indicating better model performance.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{72}$$

- sMAPE (Symmetric Mean Absolute Percentage Error): This metric expresses errors as a percentage of actual values, making it scale-independent. To reduce the asymmetry between over- and under-estimation, it is normalized by the sum of predicted and actual values, making it more robust when actual values are small.

$$\text{sMAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2} \times 100\% \tag{73}$$

[2] The code of our implementation is made open-source and is accessible at Github.com/MaherDissem/PersFL-LoadForecast.

- R^2 (Coefficient of Determination): Measures how well the predicted values match the actual data by comparing the model's performance to that of a naive prediction (a horizontal line). A score of 0 means the model's predictions are equivalent to using the mean of the observed data, while a score of 1 indicates a perfect fit. A negative R^2 means the model performs worse than a horizontal line.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i))^2} \tag{74}$$

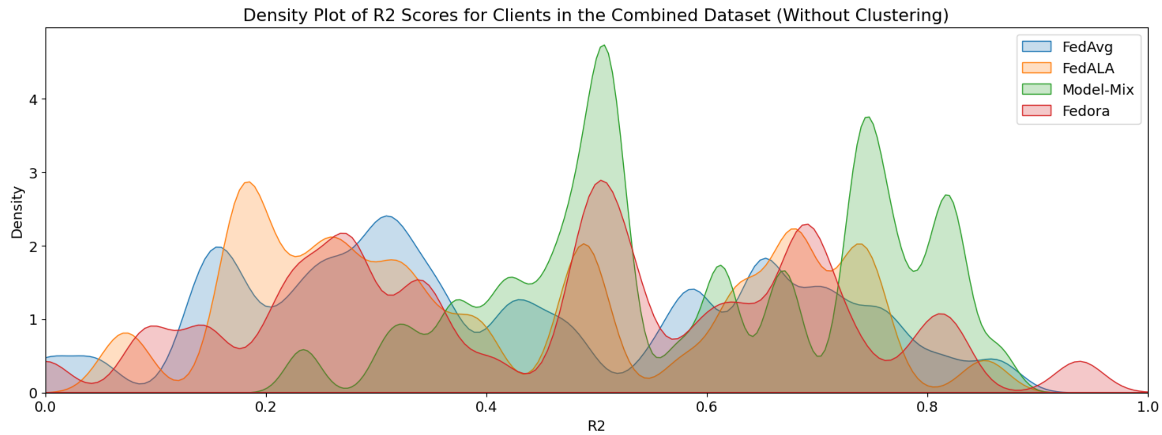
4.4 Results and Discussion

Table 4.1 presents the results achieved for each dataset under various federated learning FL training configurations. For each metric we report metrics for the 25^{th}, 50^{th} and 75^{th} quartile to provide a comprehensive view of performance across clients, from the lowest to highest performers. RMSE values are reported for the normalized time series, other metrics are scale-independent. We also illustrate and compare in Fig. 4.1 the distribution of the R^2 scores achieved by clients of the Combined dataset trained under different FL personalization settings. It is to be noted that we chose not to combine Fedora with FedCluster, as Fedora is designed to independently assess client similarity and assign low collaboration weights to dissimilar clients rather than isolating them into separate clusters

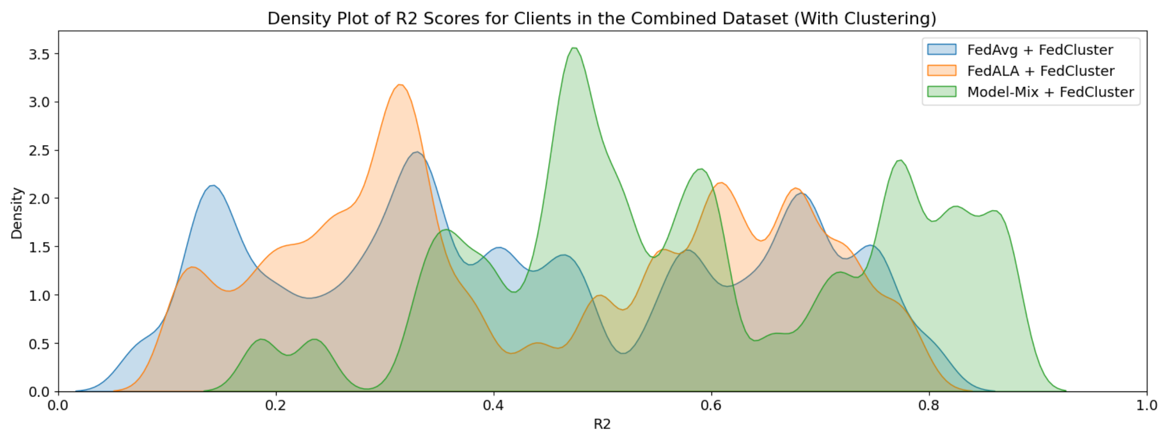
For each dataset, we observe a drastic improvement in performance when comparing isolated client training (i.e. clients individually trained on their local data) to federated learning. Specifically, after applying the FedAvg algorithm, R^2 scores increase by 70, 27, and 15 points across the 25^{th}, 50^{th} and 75^{th} quartiles, respectively for the IRISE dataset. Similarly, for the REFIT dataset, R^2 scores increase by 52, 33, and 36 points for these quartiles, and by 53, 40, and 22 points respectively, for the Combined dataset. These substantial gains can be attributed to the small size of these local datasets, which makes them insufficient for effectively training deep learning models. By leveraging the shared learning across clients, the FedAvg approach enables performance improvements regardless of data similarity, as clients benefit from the pooled insights of the federated model.

Table 4.1: Performance comparison across all training and data settings. For each dataset, metric, and quartile, the best performance is highlighted in bold.

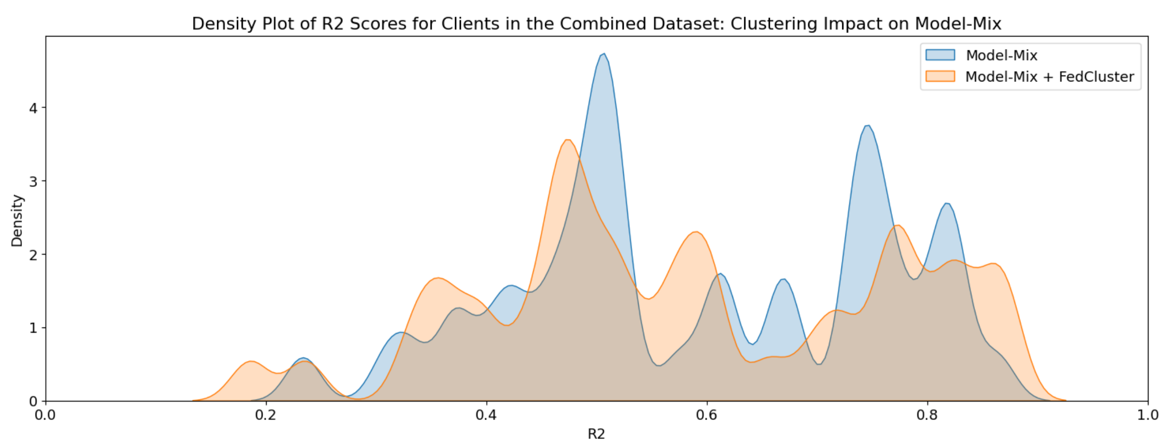
Dataset	Training method	sMAPE			RMSE			R^2		
		Q25	Q50	Q75	Q25	Q50	Q75	Q25	Q50	Q75
IRISE	Local training	95.20	75.03	66.56	0.107	0.081	0.023	-0.33	0.34	0.57
	FedAvg	63.37	47.39	42.40	0.081	0.073	0.020	0.37	0.61	0.72
	Fedora	63.62	55.72	45.86	0.089	0.063	0.013	0.47	0.63	0.70
	FedALA	63.31	48.06	41.21	0.083	0.074	0.018	0.37	0.63	0.71
	Model-Mix	59.23	48.05	40.84	0.073	**0.061**	0.017	**0.57**	0.72	**0.79**
	FedAvg + FedCluster	61.70	50.50	42.00	0.087	0.077	0.017	0.49	0.62	0.71
	FedALA + FedCluster	58.94	**46.25**	**40.94**	0.085	0.078	0.017	0.45	0.62	0.69
	Model-Mix + FedCluster	**58.45**	49.79	41.63	**0.072**	0.063	**0.014**	**0.57**	**0.74**	0.76
REFIT	Local training	113.18	100.63	84.54	0.114	0.107	0.074	-0.36	-0.09	0.01
	FedAvg	74.77	66.47	55.68	0.098	0.086	0.056	0.16	0.24	0.37
	Fedora	74.20	70.88	55.49	0.093	0.085	0.057	0.18	0.28	0.42
	FedALA	69.14	62.78	53.61	0.100	0.075	0.057	0.20	0.29	0.37
	Model-Mix	74.07	**61.06**	**47.15**	**0.053**	0.071	**0.050**	**0.39**	**0.44**	**0.54**
	FedAvg + FedCluster	**68.15**	62.52	51.67	0.097	0.075	0.057	0.21	0.26	0.36
	FedALA + FedCluster	74.60	64.54	53.56	0.096	0.082	0.064	0.20	0.27	0.34
	Model-Mix + FedCluster	75.98	66.34	49.43	0.081	**0.069**	**0.050**	0.38	0.43	0.49
Combined data	Local training	104.23	90.39	71.99	0.115	0.102	0.059	-0.28	-0.03	0.43
	FedAvg	70.36	59.14	47.54	0.092	0.076	0.048	0.25	0.37	0.65
	Fedora	70.27	57.58	46.41	0.092	0.074	0.043	0.28	0.50	0.66
	FedALA	65.52	54.50	43.20	0.093	0.080	0.046	0.24	0.39	0.65
	Model-Mix	69.63	55.12	43.67	**0.075**	**0.066**	0.038	**0.48**	**0.57**	0.75
	FedAvg + FedCluster	65.91	54.87	44.60	0.094	0.080	0.041	0.27	0.41	0.63
	FedALA + FedCluster	66.74	56.80	45.11	0.096	0.080	0.041	0.28	0.39	0.62
	Model-Mix + FedCluster	**64.40**	**50.63**	**41.88**	0.078	**0.066**	**0.036**	0.47	**0.57**	**0.76**



(a) Comparison of the PFL algorithms



(b) Comparison of the PFL algorithms when clustering is applied



(c) Comparison of the performance of the Model-Mix method with and without clustering

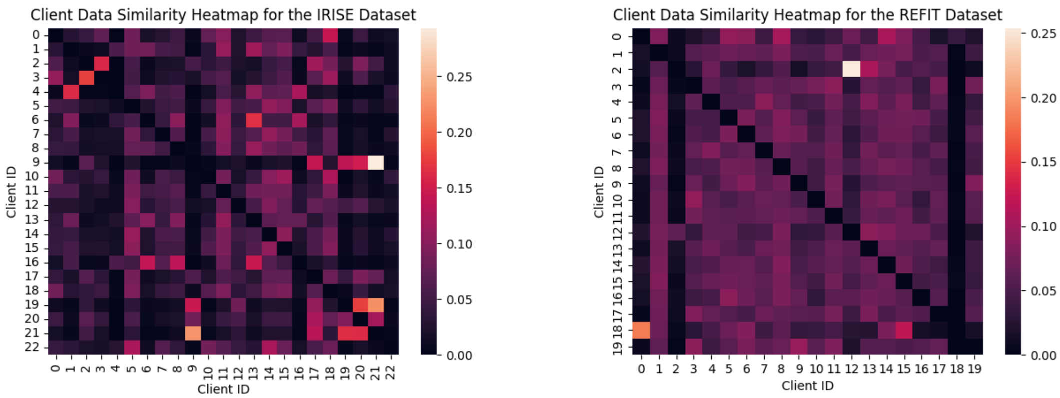
Figure 4.1: Density plots showing R^2 scores achieved by all 43 clients in the Combined dataset across various training settings.

Compared to the FedAvg baseline, the PFL algorithms Fedora and FedALA show slight performance gains across all simulation settings. However, these gains are minimal and do not significantly distinguish them from the FedAvg results.

The FedALA results suggest that the SCINet model training is robust to small variations in initial weights as despite optimizing the model's weight initialization at the start of each FL training round, SCINet ultimately converges to a similar weight distribution, resulting in consistent end performance across rounds.

Fedora adjusts parameter aggregation by prioritizing contributions from clients with more similar data, aiming to enhance the model's focus on relevant knowledge. Examining the client similarity matrix after convergence, as shown in Fig. 4.2, we find that each client assigns very low weights to most other clients, signifying that only a small subset of clients is considered similar enough to be influential. This limited interaction indicates that in this particular data context, most clients struggle to find sufficiently similar peers, leading to a reliance on a narrow range of beneficial weights from a few clients, while the majority of other, dissimilar, weights are minimally impactful. Moreover, some client can achieve good performance when focusing on a small set of similar clients, but others would benefit from collaboration with a wider range of models, even those not deemed similar enough. This is illustrated in Fig. 4.1(a) that shows that while a few clients achieve impressive performance with R^2 scores as high as 95%, others struggle significantly, attaining very low scores. To address this limitation, we experimented with restricting aggregation to only the most similar clients, but this adjustment did not improve overall performance. In conclusion, while Fedora prevents interactions that could degrade the model's learned knowledge, it also restricts exposure to broader data diversity, which limits the potential for generalization from a wider range of client data.

The model mixing approach achieves the highest forecasting accuracy, with R^2 scores reaching 57%, 72%, and 79% for the REFIT dataset; 39%, 44%, and 54% for the IRISE dataset; and 48%, 57%, and 75% for the combination of the two datasets, respectively for the 25^{th}, 50^{th} and 75^{th} quartiles. As shown in Table 4.1 and Fig. 4.1(a), this approach also results in reduced score dispersion across clients, indicating more uniform performance across the client pool, unlike other methods,



(a) Heatmap for clients of the IRISE dataset

(b) Heatmap for clients of the REFIT dataset

Figure 4.2: Heatmaps illustrating client data similarity as estimated by the Fedora PFL algorithm. Each client on the y-axis assigns a normalized similarity weight to each client on the x-axis, indicating the relative importance of other clients' data in its model aggregation process.

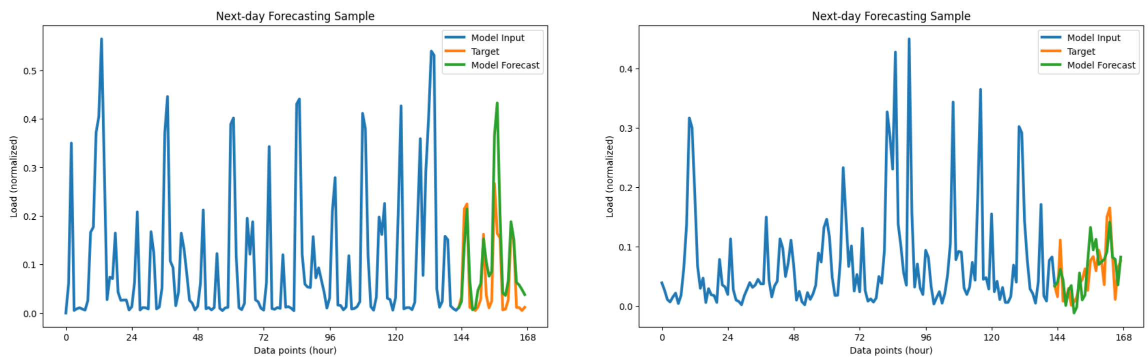
where there is a more noticeable gap between the highest and lowest performing clients. This consistency indicates that, compared to previous methods, the weight mixing technique enables clients to build models that are better suited to their specific data characteristics while still benefiting from the collective insights of the client pool. The strength of this approach lies in its dual-model structure, where each client maintains both a local and a global model connected through the weight space. This setup allows each client to establish a unique balance between the significance of their own local data and the aggregated knowledge from other clients in the federated model, resulting in an overall mixed model that is finely tuned to each client's distinct data profile.

The model clustering algorithm demonstrates superior performance in certain scenarios, particularly with the combined dataset. When paired with the model mixing personalization method, it achieves the highest score for the third quartile across all metrics, along with the lowest error for the first sMAPE quantile and the best R^2 and RMSE for the second quartile. However, in all these instances, the differences between the best and next-best approaches are notably small. To further assess these results, we conduct a paired t-test on the R^2 scores of clients in the Combined dataset trained under the Model-Mix PFL strategy with and without clustering. The analysis yielded a t-statistic of 0.81, corresponding to a p-value of 0.42, indicating that the observed improvement is indeed not statistically significant.

For a more detailed perspective, we illustrate in Fig. 4.1(c) the distribution of R^2 scores for the Model-Mix approach with and without clustering. This figure shows that, while clustering enables some clients to achieve higher R^2 scores (indicated by more clients near the 0.87 mark), it also results in a higher concentration of clients around the 20% and 35% score levels, suggesting that at the same time some other clients experience lower performance with clustering than without. Additionally, the R^2 score density plot shows distinct modes at slightly different values, indicating different areas of high concentration for each approach but with no major advantage on the global perspective. Other parts of the distribution do not consistently favor either clustering or non-clustering, suggesting a nuanced impact where clustering helps certain clients but does not yield a uniform benefit across all clients. This outcome suggests that clustering, while limiting interactions with dissimilar data to reduce negative transfer, may also hinder the flow of beneficial general knowledge to certain clusters by isolating them.

Overall, our experiments demonstrate that framing the smart building load forecasting problem within a federated learning context yields substantial benefits, significantly enhancing performance while addressing the data privacy concerns of building inhabitants. We explored various personalization approaches to further optimize the performance of the client pool by tackling the non-IID data challenge. Among these strategies, the model mixing approach proved to be the most effective. In these experiments, for this optimal approach, we end up with sMAPE values ranging from 40 to 65 and R^2 scores between 40% and 80% for clients between the first and third quantile, across all datasets. This performance considerably outperforms other methods, and in some cases, it can be improved by integrating a client clustering mechanism, though this additional performance boost is not always significant.

Despite this, the performance remains lacking for some clients and offers room for further improvement. Upon examining the client data and forecasting mechanisms, we attribute this challenge to the inherent noise and frequent, seemingly random fluctuations in the particular time series datasets we are using. These characteristics complicate the forecasting task, as models can generally capture the overall shape of the curve but struggle to accurately predict these erratic fluctuations. In Fig. 4.3, we illustrate a few next-day forecasting samples from low-performing clients to highlight the data's variability and oscillation patterns. The overall metrics reported in the table are the average of such next day forecasts for all samples in the test data of each client, aggregated over the client pool. Fig. 4.3(a) illustrate a sample from the IRISE dataset, where the forecasts achieve an sMAPE of 40.58 and a R^2 score of 30%. This example demonstrates that while the model is generally able to predict the number, shape, and position of load peaks and troughs for the following day, it struggles to precisely estimate their exact values, leading to a high error. Similarly, Fig. 4.3(b) presents a forecast from the REFIT dataset, achieving an sMAPE of 39.89 and an R^2 of 35%. Here, the model captures the overall trend of the time series but shows considerable deviations at the individual data point level.



(a) Sample next forecast for a building from the IRISE dataset, achieving performance metrics of sMAPE=40.58, RMSE=0.07, and R^2=30%

(b) Sample next forecast for a building from the REFIT dataset, achieving performance metrics of sMAPE=39.89, RMSE=0.04, and R^2=35%

Figure 4.3: Next-day forecasting samples for low-performing clients obtained under the Model-Mix PFL simulation.

Improving these forecasts could involve integrating additional features that capture variations in electricity consumption, hence providing spatial and temporal correlations for the forecasting model. For example, adding sensors within buildings to monitor variables like weather conditions or occupancy rates could help. Furthermore, if more data were available, and less granular forecasts were acceptable, resampling the time series at a lower frequency could smooth out complex patterns, simplify model training, and increase forecast accuracy.

To further enhance the overall performance, future research should focus on developing more robust time series forecasting models and optimizing data processing pipelines to facilitate the forecasting task. In this study, while we have identified the optimal personalization approach for our specific use case of smart building load forecasting, these results may not generalize to other applications with different types of data or different objectives. And ultimately, as personalized federated learning is an active field of research with frequent innovations, we encourage practitioners to to explore newer approaches and methodologies before implementation.

4.5 Conclusion

In this research, we address the challenge of load forecasting in smart buildings. Specifically, we tackle the issue of limited historical data constraining the training of accurate deep learning models. To overcome this, we frame this problem in a federated load forecasting framework that leverages data from multiple buildings while ensuring privacy by keeping the load data of each building local. However, different buildings exhibit vastly different load profiles due to unique energy consumption patterns influenced by factors such as size, location, and user behavior. This results in significant heterogeneity in the data across different buildings, leading to limited performance when a standard federated learning approach, which develops a single model for all clients, is applied. Therefore, we explore methods to personalize the global model to the specific characteristics of each building. We conduct comparative evaluation experiments across different datasets and demonstrate that FL significantly improves forecasting accuracy across all buildings, with even greater benefits from PFL, highlighting its potential to drive substantial improvements in energy efficiency. The client personalization approach we find to be optimal is based on developing both a private local model tailored to each building's specific data and a federated model that collaborates with the building network to aggregate collective knowledge into a global model. We establish a connection in weight space between the local and federated models to guide their training, enabling clients to achieve both local data focus and global collaboration. This work not only contributes to the existing knowledge on load forecasting, but also lays the groundwork for future research aimed at further improving PFL methodologies for real-world energy management. Additionally, it assesses practical existing solutions to identify those that can yield the most significant energy savings in the building sector today.

Chapter 5

Conclusion

This thesis advances the efficiency and reliability of smart buildings by tackling two core tasks: fault detection through anomaly detection and improved load forecasting. The first part addresses the need for automatic detection of faults using time series sensor data with machine learning-based anomaly detection methods. The second part focuses on enhancing energy efficiency through more accurate load forecasting, despite two common issues that can hinder the performance of deep learning models: anomalies in historical data and small training datasets for new buildings.

For anomaly detection, we identify reconstruction autoencoders as among the most effective methods. However, these models are very sensitivity to neural architecture hyperparameters and the optimal configuration varies significantly across buildings, sensor types, and anomaly characteristics, complicating parameter tuning. To address this, we propose a reinforcement learning-guided neural architecture search to automate the design of the neural network. Our experiments demonstrate that this approach outperforms traditional random search for hyperparameter tuning, and that it may be applied to optimize occupancy estimation models by detecting and filtering anomalous samples from their data.

For the second task of optimizing data quality for load forecasting, we address a common limitation in anomaly detection models: the assumption of anomaly-free training data, which is rarely realistic in practical scenarios. To overcome this, we propose a feature memory bank-based anomaly detection framework that filters anomalies during training by extracting features, collecting them in a memory bank, and removing outliers. During inference, we detect anomalies by measuring the distance of observed patterns from this collected feature set, flagging those that deviate significantly as anomalous. In the case of occupancy estimation, anomalous timestamps can be omitted from training data, as the task is classification-based and does not depend heavily on time correlations. However, for load forecasting, where temporal continuity is critical, we have to replace anomalous data points with plausible patterns. We achieve this by training a recursive denoising autoencoder on the data samples deemed anomaly-free, randomly masking values during training to teach the model to generate realistic values for them. Our experiments with two widely used forecasting models across various building datasets and use cases show that integrating anomaly detection and imputation significantly improves forecasting accuracy.

To address data scarcity in load forecasting for deep learning models, we demonstrate that federated learning significantly enhances forecasting accuracy for buildings with small datasets, compared to local training. We further improve performance by tackling the non-IID nature of smart buildings, which are influenced by factors such as building size, location, weather conditions, and occupant behavior. We conduct rigorous comparative experiments to identify the optimal personalized federated learning approach and illustrate how it leads to substantial improvements.

In sum, this research makes significant contributions to anomaly detection and load forecasting in smart buildings, offering promising prospects for improving energy efficiency, reliability, and expanding applications to other time series-based domains. However, further advancements are needed to refine these systems. While our study focuses on univariate data, exploring multivariate approaches could provide deeper insights into anomalies and more accurate forecasts by leveraging correlations across multiple sensors. Additionally, although our neural architecture search enhances model selection, there remains potential for optimizing search time. Similarly, future research in time series forecasting frameworks could further enhance their accuracy. Furthermore, another key area for future work is improving model explainability because while deep learning models are effective, their black-box nature can hinder trust and adoption. Developing interpretable models will enhance transparency, making it easier for stakeholders to understand and trust the predictions, ultimately supporting better decision-making in smart building systems.

Bibliography

[1] ABB. Energy efficiency of smart buildings: Towards zero consumption and beyond. *White paper: ABB Motion*, 2021. URL https://www.energyefficiencymovement.com/wp-content/uploads/2021/05/ABB_EE_WhitePaper_Smart-buildings_final-1.pdf. Retrieved December 4, 2023.

[2] Menatalla Abououf, Rabeb Mizouni, Shakti Singh, Hadi Otrok, and Ernesto Damiani. Self-supervised online and lightweight anomaly and event detection for iot devices. *IEEE Internet of Things Journal*, 9(24):25285–25299, 2022. doi: 10.1109/JIOT.2022.3196049.

[3] Bovas Abraham and Johannes Ledolter. *Statistical methods for forecasting*. John Wiley & Sons, 2009.

[4] Muhammad Waseem Ahmad, Monjur Mourshed, David Mundow, Mario Sisinni, and Yacine Rezgui. Building energy metering and environmental monitoring – a state-of-the-art review and directions for future research. *Energy and Buildings*, 120:85–102, 2016. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2016.03.059. URL https://www.sciencedirect.com/science/article/pii/S0378778816302158.

[5] Sadi Alawadi, David Mera, Manuel Fernández-Delgado, Fahed Alkhabbas, Carl Magnus Olsson, and Paul Davidsson. A comparison of machine learning algorithms for forecasting indoor temperature in smart buildings. *Energy Systems*, pages 1–17, 2020.

[6] Andres M. Alonso, Francisco J. Nogales, and Carlos Ruiz. Hierarchical clustering for smart meter electricity loads based on quantile autocovariances. *IEEE Transactions on Smart Grid*,

11(5):4522–4530, sep 2020. doi: 10.1109/tsg.2020.2991316. URL https://doi.org/10.1109%2Ftsg.2020.2991316.

[7] Majed A Alotaibi. Machine learning approach for short-term load forecasting using deep neural network. *Energies*, 15(17):6261, 2022.

[8] Manar Amayri, Abhay Arora, Stephane Ploix, Sanghamitra Bandhyopadyay, Quoc-Dung Ngo, and Venkata Ramana Badarla. Estimating occupancy in heterogeneous sensor environment. *Energy and Buildings*, 129:46–58, 2016. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2016.07.026. URL https://www.sciencedirect.com/science/article/pii/S0378778816306223.

[9] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[10] Pandarasamy Arjunan, Harshad D. Khadilkar, Tanuja Ganu, Zainul M. Charbiwala, Amarjeet Singh, and Pushpendra Singh. Multi-user energy consumption monitoring and anomaly detection with partial context information. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, BuildSys '15, page 35–44, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450339810. doi: 10.1145/2821650.2821662. URL https://doi.org/10.1145/2821650.2821662.

[11] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017. doi: 10.1109/MSP.2017.2743240.

[12] Abdul Azeem, Idris Ismail, Syed Muslim Jameel, and Vivekananda Rajah Harindran. Electrical load forecasting models for different generation modalities: a review. *IEEE Access*, 9: 142239–142263, 2021.

[13] Donald Azuatalam, Wee-Lih Lee, Frits de Nijs, and Ariel Liebman. Reinforcement learning for whole-building hvac control and demand response. *Energy and AI*, 2:100020, 2020.

ISSN 2666-5468. doi: https://doi.org/10.1016/j.egyai.2020.100020. URL https://www.sciencedirect.com/science/article/pii/S2666546820300203.

[14] Abdulrahman Ba-Alawi, Jorge Loy-Benitez, SangYun Kim, and Changkyoo Yoo. Missing data imputation and sensor self-validation towards a sustainable operation of wastewater treatment plants via deep variational residual autoencoders. *Chemosphere*, 288:132647, 10 2021. doi: 10.1016/j.chemosphere.2021.132647.

[15] Jayme Garcia Arnal Barbedo. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Computers and Electronics in Agriculture*, 153:46–53, 2018. ISSN 0168-1699. doi: https://doi.org/10.1016/j.compag.2018.08.013. URL https://www.sciencedirect.com/science/article/pii/S0168169918304617.

[16] Kaustav Basu. *Classification Techniques for Non-intrusive Load Monitoring and Prediction of Residential Loads*. PhD thesis, Université de Grenoble, 11 2014.

[17] Gustavo E. A. P. A. Batista and Maria Carolina Monard. A study of k-nearest neighbour as an imputation method. In *International Conference on Health Information Science*, 2002. URL https://api.semanticscholar.org/CorpusID:37493644.

[18] Alessandro Beghi, Luca Cecchinato, Mirco Rampazzo, and Francesco Simmini. Load forecasting for the efficient energy management of hvac systems. In *2010 IEEE International Conference on Sustainable Energy Technologies (ICSET)*, pages 1–6, 2010. doi: 10.1109/ICSET.2010.5684414.

[19] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Mach. Learn.*, 79 (1–2):151–175, May 2010. ISSN 0885-6125. doi: 10.1007/s10994-009-5152-4. URL https://doi.org/10.1007/s10994-009-5152-4.

[20] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[21] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[22] Jinsung Byun, Insung Hong, Byeongkwan Kang, and Sehyun Park. A smart energy distribution and management system for renewable energy distribution and context-aware services based on user patterns and load forecasting. *IEEE Transactions on Consumer Electronics*, 57 (2):436–444, 2011. doi: 10.1109/TCE.2011.5955177.

[23] Vitor Cerqueira, Luis Torgo, and Carlos Soares. Machine learning vs statistical methods for time series forecasting: Size matters, 2019. URL https://arxiv.org/abs/1909.13316.

[24] Cristian Challu, Peihong Jiang, Ying Nian Wu, and Laurent Callot. Deep generative model with hierarchical latent factors for time series anomaly detection, 2022. URL https://arxiv.org/abs/2202.07586.

[25] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi, editors, *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012. URL https://aclanthology.org/W14-4012.

[26] Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines. *IEEE Access*, 9:120043–120065, 2021. doi: 10.1109/ACCESS.2021.3107975.

[27] Wonjun Choi and Sangwon Lee. Performance evaluation of deep learning architectures for load and temperature forecasting under dataset size constraints and seasonality. *Energy and Buildings*, 288:113027, 2023. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2023.

113027. URL https://www.sciencedirect.com/science/article/pii/S0378778823002578.

[28] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search, 2021.

[29] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. URL http://arxiv.org/abs/1412.3555. cite arxiv:1412.3555Comment: Presented in NIPS 2014 Deep Learning and Representation Learning Workshop.

[30] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2020. doi: 10.1109/JIOT.2019.2958185.

[31] Sarah Darby. The effectiveness of feedback on energy consumption. *A Review for DEFRA of the Literature on Metering, Billing and direct Displays*, 486, 01 2006.

[32] Jan G De Gooijer and Rob J Hyndman. 25 years of time series forecasting. *International journal of forecasting*, 22(3):443–473, 2006.

[33] Chirag Deb, Fan Zhang, Junjing Yang, Siew Eang Lee, and Kwok Wei Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74:902–924, 2017. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2017.02.085. URL https://www.sciencedirect.com/science/article/pii/S1364032117303155.

[34] Thomas Defard, Aleksandr Setkov, Angélique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *ICPR Workshops*, 2020. URL https://api.semanticscholar.org/CorpusID:226976039.

[35] Xianzhong Ding, Wan Du, and Alberto Cerpa. Octopus: Deep reinforcement learning

for holistic smart building control. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '19, page 326–335, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450370059. doi: 10.1145/3360322.3360857. URL https://doi.org/10.1145/3360322.3360857.

[36] Maher Dissem and Manar Amayri. Unsupervised anomaly detection and imputation in noisy time series data for enhancing load forecasting. *Applied Intelligence*, 55(1):11, Nov 2024. ISSN 1573-7497. doi: 10.1007/s10489-024-05856-6. URL https://doi.org/10.1007/s10489-024-05856-6.

[37] Maher Dissem and Manar Amayri. Towards efficient federated load forecasting: Personalization mechanisms and their impact. *IEEE Internet of Things Journal*, submitted.

[38] Maher Dissem, Manar Amayri, and Nizar Bouguila. Neural architecture search for anomaly detection in time-series data of smart buildings: A reinforcement learning approach for optimal autoencoder design. *IEEE Internet of Things Journal*, 11(10):18059–18073, 2024. doi: 10.1109/JIOT.2024.3360882.

[39] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially no barriers in neural network energy landscape, 2019. URL https://arxiv.org/abs/1803.00885.

[40] M. Economidou, V. Todeschi, P. Bertoldi, D. D'Agostino, P. Zangheri, and L. Castellazzi. Review of 50 years of eu energy efficiency policies for buildings. *Energy and Buildings*, 225:110322, 2020. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2020.110322. URL https://www.sciencedirect.com/science/article/pii/S0378778820317229.

[41] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[42] Chenguang Fang and Chen Wang. Time series data imputation: A survey on deep learning approaches, 2020.

[43] Chenguang Fang, Shaoxu Song, Zhiwei Chen, and Acan Gui. Fine-grained fuel consumption prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 2783–2791, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3357836. URL https://doi.org/10.1145/3357384.3357836.

[44] Behnam Farsi, Manar Amayri, Nizar Bouguila, and Ursula Eicker. On short-term load forecasting using machine learning techniques and a novel parallel deep lstm-cnn approach. *IEEE Access*, 9:31191–31212, 2021. doi: 10.1109/ACCESS.2021.3060290.

[45] Romain Fontugne, Jorge Ortiz, Nicolas Tremblay, Pierre Borgnat, Patrick Flandrin, Kensuke Fukuda, David Culler, and Hiroshi Esaki. Strip, bind, and search: A method for identifying abnormal energy consumption in buildings. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, IPSN '13, page 129–140, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450319591. doi: 10.1145/2461381.2461399. URL https://doi.org/10.1145/2461381.2461399.

[46] Guanyu Gao, Jie Li, and Yonggang Wen. Deepcomfort: Energy-efficient thermal comfort control in buildings via reinforcement learning. *IEEE Internet of Things Journal*, 7(9):8472–8484, 2020. doi: 10.1109/JIOT.2020.2992117.

[47] Yuan Gao, Yingjun Ruan, Chengkuan Fang, and Shuai Yin. Deep learning and transfer learning models of energy consumption forecasting for a building with poor information data. *Energy and Buildings*, 223:110156, 2020. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2020.110156. URL https://www.sciencedirect.com/science/article/pii/S0378778819324843.

[48] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns, 2018. URL https://arxiv.org/abs/1802.10026.

[49] John Graham. Missing data analysis: Making it work in the real world. *Annual review of psychology*, 60:549–76, 01 2009. doi: 10.1146/annurev.psych.58.110405.085530.

[50] S. Hadri, Y. Naitmalek, M. Najib, M. Bakhouya, Y. Fakhri, and M. Elaroussi. A comparative study of predictive approaches for load forecasting in smart buildings. *Procedia Computer Science*, 160:173–180, 2019. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2019.09.458. URL https://www.sciencedirect.com/science/article/pii/S1877050919316734. The 10th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2019) / The 9th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2019) / Affiliated Workshops.

[51] Seok-Ju Hahn, Minwoo Jeong, and Junghye Lee. Connecting low-loss subspace for personalized federated learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 505–515, 2022.

[52] Yu He, Fengji Luo, Mingyang Sun, and Gianluca Ranzi. Privacy-preserving and hierarchically federated framework for short-term residential load forecasting. *IEEE Transactions on Smart Grid*, 14(6):4409–4423, 2023. doi: 10.1109/TSG.2023.3268633.

[53] Siamak Hoseinzadeh, Daniele Groppi, Adriana Scarlet Sferra, Umberto Di Matteo, and Davide Astiaso Garcia. The prismi plus toolkit application to a grid-connected mediterranean island. *Energies*, 15(22), 2022. ISSN 1996-1073. doi: 10.3390/en15228652. URL https://www.mdpi.com/1996-1073/15/22/8652.

[54] Tianyu Hu, Qinglai Guo, Xinwei Shen, Rongli Wu, and Haoning Xi. Utilizing unlabeled data to detect electricity fraud in ami: A semisupervised deep learning approach. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–13, 01 2019. doi: 10.1109/TNNLS.2018.2890663.

[55] Chengqiang Huang, Yulei Wu, Yuan Zuo, Ke Pei, and Geyong Min. Towards experienced anomaly detector through reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

[56] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25566-3.

[57] Energy Institute. Statistical review of world energy, 2023.

[58] Mengda Jia, Ali Komeily, Yueren Wang, and Ravi S. Srinivasan. Adopting internet of things for the development of smart buildings: A review of enabling technologies and applications. *Automation in Construction*, 101:111–126, 2019. ISSN 0926-5805. doi: https://doi.org/10.1016/j.autcon.2019.01.023. URL https://www.sciencedirect.com/science/article/pii/S0926580518307064.

[59] Xi Jiang, Jianlin Liu, Jinbao Wang, Qiang Nie, Kai WU, Yong Liu, Chengjie Wang, and Feng Zheng. Softpatch: Unsupervised anomaly detection with noisy data. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 15433–15445. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/637a456d89289769ac1ab29617ef7213-Paper-Conference.pdf.

[60] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

[61] Devinder Kaur, Shama Naz Islam, Md Apel Mahmud, Md Enamul Haque, and Zhao Yang Dong. Energy forecasting in smart grid systems: recent advancements in probabilistic deep learning. *IET Generation, Transmission & Distribution*, 16(22):4461–4479, 2022.

[62] Jack Kelly and William Knottenbelt. The uk-dale dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes. *Scientific Data*, 2(1), March 2015. ISSN 2052-4463. doi: 10.1038/sdata.2015.7. URL http://dx.doi.org/10.1038/sdata.2015.7.

[63] Sinnesbichler H Kersken M. Simulation study on the energy saving potential of a heating control system featuring presence detection and weather forecasting. *Internal report of Fraunhofer Institute*, 2013.

[64] Tommie Kerssies. Neural architecture search for visual anomaly segmentation. *ArXiv*, abs/2304.08975, 2023. URL https://api.semanticscholar.org/CorpusID:258187082.

[65] T Kober, H-W Schiffer, M Densing, and E Panos. Global energy perspectives to 2060–wec's world energy scenarios 2019. *Energy Strategy Reviews*, 31:100523, 2020.

[66] D. Kolokotsa, D. Rovas, E. Kosmatopoulos, and K. Kalaitzakis. A roadmap towards intelligent net zero- and positive-energy buildings. *Solar Energy*, 85(12):3067–3084, 2011. ISSN 0038-092X. doi: https://doi.org/10.1016/j.solener.2010.09.001. URL https://www.sciencedirect.com/science/article/pii/S0038092X10002847. Progress in Solar Energy 2.

[67] Weicong Kong, Zhao Yang Dong, Youwei Jia, David J. Hill, Yan Xu, and Yuan Zhang. Short-term residential load forecasting based on lstm recurrent neural network. *IEEE Transactions on Smart Grid*, 10(1):841–851, 2019. doi: 10.1109/TSG.2017.2753802.

[68] Timilehin Labeodan, Wim Zeiler, Gert Boxem, and Yang Zhao. Occupancy measurement in commercial office buildings for demand-driven control applications—a survey and detection system evaluation. *Energy and Buildings*, 93:303–314, 2015. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2015.02.028. URL https://www.sciencedirect.com/science/article/pii/S0378778815001334.

[69] Dasom Lee and David J. Hess. Data privacy and residential smart meters: Comparative analysis and harmonization potential. *Utilities Policy*, 70:101188, 2021. ISSN 0957-1787. doi: https://doi.org/10.1016/j.jup.2021.101188. URL https://www.sciencedirect.com/science/article/pii/S0957178721000229.

[70] Jinbo Li, Hesam Izakian, Witold Pedrycz, and Iqbal Jamal. Clustering-based anomaly detection in multivariate time series data. *Applied Soft Computing*, 100:106919, 2021.

ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2020.106919. URL https://www.sciencedirect.com/science/article/pii/S1568494620308577.

[71] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 367–377. PMLR, 22–25 Jul 2020. URL https://proceedings.mlr.press/v115/li20c.html.

[72] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[73] Yuening Li, Zhengzhang Chen, Daochen Zha, Kaixiong Zhou, Haifeng Jin, Haifeng Chen, and Xia Hu. Autood: Automated outlier detection via curiosity-guided search and self-imitation learning. *CoRR*, abs/2006.11321, 2020. URL https://arxiv.org/abs/2006.11321.

[74] Yuening Li, Daochen Zha, Praveen Venugopal, Na Zou, and Xia Hu. Pyodds: An end-to-end outlier detection system with automated machine learning. In *Companion Proceedings of the Web Conference 2020*, pages 153–157, 2020.

[75] Antonio Liguori, Romana Markovic, Thi Thu Ha Dam, Jérôme Frisch, Christoph van Treeck, and Francesco Causone. Indoor environment data time-series reconstruction using autoencoder neural networks. *Building and Environment*, 191:107623, 2021. ISSN 0360-1323. doi: https://doi.org/10.1016/j.buildenv.2021.107623. URL https://www.sciencedirect.com/science/article/pii/S0360132321000354.

[76] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[77] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2351–2363. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/18df51b97ccd68128e994804f3eccc87-Paper.pdf.

[78] Weixuan Lin, Di Wu, and Benoit Boulet. Spatial-temporal residential short-term load forecasting via graph neural networks. *IEEE Transactions on Smart Grid*, 12(6):5373–5384, 2021. doi: 10.1109/TSG.2021.3093515.

[79] Minhao LIU, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia LAI, Lingna Ma, and Qiang Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 5816–5828. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/266983d0949aed78a16fa4782237dea7-Paper-Conference.pdf.

[80] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022.

[81] Xiaolei Liu, Zi Lin, and Ziming Feng. Short-term offshore wind speed forecast by seasonal arima - a comparison against gru and lstm. *Energy*, 227:120492, 2021. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2021.120492. URL https://www.sciencedirect.com/science/article/pii/S0360544221007416.

[82] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M. Shamim Hossain. Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal*, 8(8):6348–6358, 2021. doi: 10.1109/JIOT.2020.3011726.

[83] Gustavo Felipe Martin Nascimento, Frederic Wurtz, Patrick Kuo-Peng, Benoit Delinchant, Nelson Batistela, and Tiansi Laranjeira. Green-er–electricity consumption data of a tertiary building. *Frontiers in Sustainable Cities*, 5, 08 2023. doi: 10.3389/frsc.2023.1043657.

[84] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2016. URL https://api.semanticscholar.org/CorpusID:14955348.

[85] David Murray, Lina Stankovic, and Vladimir Stankovic. An electrical load measurements dataset of united kingdom households from a two-year longitudinal study. *Scientific data*, 4 (1):1–12, 2017.

[86] Srinarayana Nagarathinam, Vishnu Menon, Arunchandar Vasan, and Anand Sivasubramaniam. Marco - multi-agent reinforcement learning based control of building hvac systems. In *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*, e-Energy '20, page 57–67, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380096. doi: 10.1145/3396851.3397694. URL https://doi.org/10.1145/3396851.3397694.

[87] Alfredo Nespoli, Emanuele Ogliari, Silvia Pretto, Michele Gavazzeni, Sonia Vigani, and Franco Paccanelli. Electrical load forecast by means of lstm: The impact of data quality. *Forecasting*, 3(1):91–101, 2021.

[88] Hussain Nizam, Samra Zafar, Zefeng Lv, Fan Wang, and Xiaopeng Hu. Real-time deep anomaly detection framework for multivariate time-series data in industrial iot. *IEEE Sensors Journal*, 22(23):22836–22849, 2022. doi: 10.1109/JSEN.2022.3211874.

[89] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, pages 3878–3887. PMLR, 2018.

[90] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing, 2018.

[91] Hojjat Rakhshani, Hassan Ismail Fawaz, Lhassane Idoumghar, Germain Forestier, Julien Lepagnot, Jonathan Weber, Mathieu Brévilliers, and Pierre-Alain Muller. Neural architecture search for time series classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020. doi: 10.1109/IJCNN48605.2020.9206721.

[92] Mauro Ribeiro, Katarina Grolinger, Hany F. ElYamany, Wilson A. Higashino, and Miriam A.M. Capretz. Transfer learning with seasonal and trend adjustment for cross-building energy forecasting. *Energy and Buildings*, 165:352–363, 2018. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2018.01.034. URL https://www.sciencedirect.com/science/article/pii/S0378778817329171.

[93] Vadim Romanuke. Time series smoothing improving forecasting. *Applied Computer Systems*, 26(1):60–70, 2021.

[94] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14298–14308, 2021. URL https://api.semanticscholar.org/CorpusID:235436036.

[95] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14318–14328, 2022.

[96] Muhammad Sajjad, Zulfiqar Ahmad Khan, Amin Ullah, Tanveer Hussain, Waseem Ullah, Mi Young Lee, and Sung Wook Baik. A novel cnn-gru-based hybrid approach for short-term residential load forecasting. *IEEE Access*, 8:143759–143768, 2020. doi: 10.1109/ACCESS.2020.3009537.

[97] M Santamouris and K Vasilakopoulou. Present and future energy consumption of buildings: Challenges and opportunities towards decarbonisation. *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, 1:100002, 2021.

[98] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: a comprehensive evaluation. *Proc. VLDB Endow.*, 15(9):1779–1797, may 2022. ISSN

2150-8097. doi: 10.14778/3538598.3538602. URL https://doi.org/10.14778/3538598.3538602.

[99] Vikash Sehwag, Arjun Nitin Bhagoji, Liwei Song, Chawin Sitawarin, Daniel Cullina, Mung Chiang, and Prateek Mittal. Analyzing the robustness of open-world machine learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 105–116, 2019.

[100] Rajat Sen, Hsiang-Fu Yu, and Inderjit Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting, 2019.

[101] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=H1aIuk-RW.

[102] Arnaldo Sgueglia, Andrea Di Sorbo, Corrado Aaron Visaggio, and Gerardo Canfora. A systematic literature review of iot time series anomaly detection solutions. *Future Generation Computer Systems*, 134:170–186, 2022. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2022.04.005. URL https://www.sciencedirect.com/science/article/pii/S0167739X22001285.

[103] Samarth Sinha, Han Zhang, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, and Augustus Odena. Small-GAN: Speeding up GAN training using core-sets. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9005–9015. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/sinha20b.html.

[104] Andrew J. Sonta, Perry E. Simmons, and Rishee K. Jain. Understanding building occupant activities at scale: An integrated knowledge-based and data-driven approach. *Advanced Engineering Informatics*, 37:1–13, 2018. ISSN 1474-0346. doi: https://doi.org/10.1016/j.aei.2018.04.009. URL https://www.sciencedirect.com/science/article/pii/S147403461730410X.

[105] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks*, 129:249–260, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.06.005. URL https://www.sciencedirect.com/science/article/pii/S0893608020302185.

[106] John Stankovic. Research directions for the internet of things. *Internet of Things Journal, IEEE*, 1:3–9, 02 2014. doi: 10.1109/JIOT.2014.2312291.

[107] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2828–2837, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330672. URL https://doi.org/10.1145/3292500.3330672.

[108] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in neural information processing systems*, 33:21394–21405, 2020.

[109] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):9587–9603, 2023. doi: 10.1109/TNNLS.2022.3160699.

[110] Viet Tra, Manar Amayri, and Nizar Bouguila. Unsupervised outlier detection using neural network-based mixtures of probabilistic principal component analyzers for building chiller fault diagnosis. *Building and Environment*, 225:109620, 2022. ISSN 0360-1323. doi: https://doi.org/10.1016/j.buildenv.2022.109620. URL https://www.sciencedirect.com/science/article/pii/S0360132322008502.

[111] Marian Turowski, Moritz Weber, Oliver Neumann, Benedikt Heidrich, Kaleb Phipps, Hüseyin K. Çakmak, Ralf Mikut, and Veit Hagenmeyer. Modeling and generating synthetic anomalies for energy and power time series. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, e-Energy '22, page 471–484, New York, NY,

USA, 2022. Association for Computing Machinery. ISBN 9781450393973. doi: 10.1145/3538637.3539760. URL https://doi.org/10.1145/3538637.3539760.

[112] Roberta Vadruccio, Chiara Siragusa, and Angela Tumino. Increasing energy efficiency in smart building through internet of things retrofitting intervention. *Procedia Computer Science*, 219:263–270, 2023. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2023.01.289. URL https://www.sciencedirect.com/science/article/pii/S1877050923002971. CENTERIS – International Conference on ENTERprise Information Systems / ProjMAN – International Conference on Project MANagement / HCist – International Conference on Health and Social Care Information Systems and Technologies 2022.

[113] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. A review on the long short-term memory model. *Artificial Intelligence Review*, 53, 12 2020. doi: 10.1007/s10462-020-09838-1.

[114] Gerard Jacques van Wyk and Anna Sergeevna Bosman. Evolutionary neural architecture search for image restoration. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019. doi: 10.1109/IJCNN.2019.8852417.

[115] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[116] Lan Wang, Eric Wai Ming Lee, Richard K.K. Yuen, and Wei Feng. Cooling load forecasting-based predictive optimisation for chiller plants. *Energy and Buildings*, 198:261–274, 2019. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2019.06.016. URL https://www.sciencedirect.com/science/article/pii/S037877881930636X.

[117] Long Wang, Yong Ding, Till Riedel, Andrei Miclaus, and Michael Beigl. Data analysis on building load profiles: A stepping stone to future campus. In *2017 International Smart Cities Conference (ISC2)*, pages 1–4, 2017. doi: 10.1109/ISC2.2017.8090823.

[118] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

[119] Mitchell Wortsman, Maxwell Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. Learning neural network subspaces, 2021. URL https://arxiv.org/abs/2102.10472.

[120] Di Wu and Weixuan Lin. Efficient residential electric load forecasting via transfer learning and graph neural networks. *IEEE Transactions on Smart Grid*, 14(3):2423–2431, 2022.

[121] Di Wu and Weixuan Lin. Efficient residential electric load forecasting via transfer learning and graph neural networks. *IEEE Transactions on Smart Grid*, 14(3):2423–2431, 2023. doi: 10.1109/TSG.2022.3208211.

[122] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

[123] Jun Wu, Wenxuan Bao, Elizabeth Ainsworth, and Jingrui He. Personalized federated learning with parameter propagation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 2594–2605, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599464. URL https://doi.org/10.1145/3580305.3599464.

[124] Renjie Wu and Eamonn J. Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress (extended abstract). In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1479–1480, 2022. doi: 10.1109/ICDE53745.2022.00116.

[125] Tong Wu and Jorge Ortiz. Towards adaptive anomaly detection in buildings with deep reinforcement learning. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '19, page

380–382, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450370059. doi: 10.1145/3360322.3361011. URL https://doi.org/10.1145/3360322.3361011.

[126] Tong Wu and Jorge Ortiz. RLAD: time series anomaly detection through reinforcement learning and active learning. *CoRR*, abs/2104.00543, 2021. URL https://arxiv.org/abs/2104.00543.

[127] Xianze Xu and Zhaorui Meng. A hybrid transfer learning model for short-term electric load forecasting. *Electrical engineering*, 102(3):1371–1381, 2020.

[128] Yahoo! Research. Yahoo! Webscope dataset ydata-labeled-time-series-anomalies-v1_0. http://labs.yahoo.com/Academic_Relations, 2015.

[129] Rui Ye and Qun Dai. Implementing transfer learning across different datasets for time series forecasting. *Pattern Recognition*, 109:107617, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107617. URL https://www.sciencedirect.com/science/article/pii/S0031320320304209.

[130] Chunyong Yin, Sun Zhang, Jin Wang, and Naixue Xiong. Anomaly detection based on convolutional recurrent autoencoder for iot time series. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52:112–122, 01 2022. doi: 10.1109/TSMC.2020.2968516.

[131] Sook-Chin Yip, Wooi-Nee Tan, ChiaKwang Tan, Ming-Tao Gan, and KokSheik Wong. An anomaly detection framework for identifying energy theft and defective meters in smart grids. *International Journal of Electrical Power & Energy Systems*, 101:189–203, 2018. ISSN 0142-0615. doi: https://doi.org/10.1016/j.ijepes.2018.03.025. URL https://www.sciencedirect.com/science/article/pii/S0142061517318719.

[132] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications, 2020.

[133] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series

forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[134] Aoqian Zhang, Shaoxu Song, and Jianmin Wang. Sequential data cleaning: A statistical approach. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, page 909–924, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450335317. doi: 10.1145/2882903.2915233. URL https://doi.org/10.1145/2882903.2915233.

[135] Aoqian Zhang, Shaoxu Song, Jianmin Wang, and Philip S. Yu. Time series data cleaning: from anomaly detection to anomaly repairing. *Proc. VLDB Endow.*, 10(10):1046–1057, jun 2017. ISSN 2150-8097. doi: 10.14778/3115404.3115410. URL https://doi.org/10.14778/3115404.3115410.

[136] Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. Fedala: adaptive local aggregation for personalized federated learning. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i9.26330. URL https://doi.org/10.1609/aaai.v37i9.26330.

[137] Jianye Zhang and Peng Yin. Multivariate time series missing data imputation using recurrent denoising autoencoder. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 760–764, 2019. doi: 10.1109/BIBM47256.2019.8982996.

[138] Peihai Zhao, Xiaoyan Chang, and Mimi Wang. A novel multivariate time-series anomaly detection approach using an unsupervised deep neural network. *IEEE Access*, 9:109025–109041, 2021. doi: 10.1109/ACCESS.2021.3101844.

[139] Qiang Zhao, Zheng Chang, and Geyong Min. Anomaly detection and classification of household electricity data: A time window and multilayer hierarchical network approach. *IEEE Internet of Things Journal*, 9(5):3704–3716, 2022. doi: 10.1109/JIOT.2021.3098735.

[140] Yue Zhao and Leman Akoglu. Towards unsupervised hpo for outlier detection. *arXiv preprint arXiv:2208.11727*, 2022.

[141] Yue Zhao, Ryan Rossi, and Leman Akoglu. Automatic unsupervised outlier model selection. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 4489–4502. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/23c894276a2c5a16470e6a31f4618d73-Paper.pdf.

[142] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[143] Kaile Zhou, Dingding Hu, Rong Hu, and Jiong Zhou. High-resolution electric power load data of an industrial park with multiple types of buildings in china. *Scientific Data*, 10, 12 2023. doi: 10.1038/s41597-023-02786-9.

[144] Yingjie Zhou, Xucheng Song, Yanru Zhang, Fanxing Liu, Ce Zhu, and Lingqiao Liu. Feature encoding with autoencoders for weakly supervised anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2454–2465, June 2022. ISSN 2162-2388. doi: 10.1109/tnnls.2021.3086137. URL http://dx.doi.org/10.1109/TNNLS.2021.3086137.

[145] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. URL http://arxiv.org/abs/1611.01578.

[146] Kasim Zor and Kurtuluş Buluş. A benchmark of gru and lstm networks for short-term electric load forecasting. In *2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 598–602. IEEE, 2021.