AI-Powered Time Series Forecasting Frameworks For Building Energy Management Systems

Omar Bouhamed

A Thesis
in
The Department
of
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

June 2022

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Omar Bouhamed**
Entitled: **AI-Powered Time Series Forecasting Frameworks For Building Energy Management Systems**

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
Dr. Arash Mohammadi

_____ Examiner
Dr. Zachary Patterson

_____ Supervisor
Dr. Nizar Bouguila

_____ Co-supervisor
Dr. Manar Amayri

Approved by _____
Dr. Mohammed Mannan, Graduate Program Director

_____ 2022

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

AI-Powered Time Series Forecasting Frameworks For Building Energy Management Systems

Omar Bouhamed

Energy, as an essential aspect for socioeconomic growth, has remained an intriguing issue for many researchers worldwide. The rising need for energy drives academics and researchers to develop novel solutions for improved energy utilisation. The main contribution of this thesis is two-fold. First, we introduce an occupancy prediction framework to improve the efficiency of energy management systems in smart buildings. Occupancy prediction heavily depends on historical occupancy-related data collected from various sensor sources. Unfortunately, a major problem in that context is the difficulty to collect training data. This situation inspired us to rethink the occupancy prediction problem, proposing the use of an original principled approach based on occupancy estimation via interactive learning to collect the needed training data. Following that, the collected data, along with various features, was fed into several algorithms to predict future occupancy. This fold mainly proposes a weakly supervised occupancy prediction framework based on office sensor readings and occupancy estimations derived from an interactive learning approach. Two studies are the main emphasis of this part. The first is the prediction of three occupancy states, referred to as discrete states: absence, presence of one occupant, and presence of more than one occupant. The purpose of the second study is to anticipate the future number of occupants, i.e, continuous states. Extensive simulations were run to demonstrate the merits of the proposed prediction framework's performance and to validate the interactive learning-based approach's ability to contribute to the achievement of effective occupancy prediction.

Second, given that an accurate electric power load forecasting framework is critical for power utility companies as it increases control over the relevant infrastructure, resulting in significant improvements in energy management and scheduling, we propose an encoder-decoder model that takes advantage of the expressiveness of transformer-based encoders to produce probabilistic forecasts. Two real-world datasets are utilized to incorporate the performance of the proposed framework on two different types of data: hourly load data from the power supply company of the city of Johor in Malaysia and hourly load consumption data from one of Grenoble Institute of Technology's buildings. The former represents aggregated data, which makes identifying patterns and trends easier, but the latter was taken from a single building (non-aggregated), which increases the difficulty of forecasts. The model's performance is discussed across multiple time horizons, including 24-hour, 1-week, and 1-month predictions. The framework achieved notable improvements compared to the used baseline, Amazon DeepAr, where accuracy was improved from 87.2 percent to 96.2 percent for Malaysian data and from 52.3 percent to 68.2 percent for Grenoble data for 24 hours ahead forecasting, from 84.7 percent to 89.7 percent for Malaysian data, and from 45.5 percent to 57.2 percent for Grenoble data for 1 month ahead forecasting.

Dedicated,

To you, my inspiring parents,

To my sister,

To all those I love,

for being the pillows, role models,

cheerleading squad and sounding boards

I have needed.

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisor *Prof. Nizar Bouguila*, for the continuous support, for his patience, motivation, and immense knowledge. His guidance helped me to present a work that I am proud of. It is very difficult to put into words to exactly express my appreciation to him. I could not ask for a better advisor and mentor for my thesis.

I would like to offer my special thank to my co-supervisor *Dr. Manar Amayri* for her precious advice and support.

I would like to thank Fatma, Omar, Kamal, Ons, Ahmed, and other lab members, who have always shared their vast knowledge as well as taken the time and effort to explain various concepts in order to make sure I could thoroughly understand them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Time-series Forecasting

It is interesting to look back at the evolution of demand in recent years, as well as the increasingly competitive global economy [1]. The never-ending growth in demand has piqued the interest of researchers and businesses in time series forecasting. A time series is a collection of observations in a chronological sequence that can be as simple as a sensor readings or as complex as the power consumption of an entire city. Time series forecasting is an approach for predicting future events by analyzing prior patterns and assuming that future trends would be similar to the historical ones. In other words, forecasting is the technique of predicting future values by fitting models to past data. It involves establishing models based on historical data and using them to make observations and guide future strategic decisions [2]. Today, industries ranging from healthcare and finance to energy and retail rely on time series forecasting to assess potential risk over time [3], predict product demand [4], allocate resources [5], and a variety of other applications.

The primary focus of this thesis is on energy and buildings. Energy is becoming a major economic problem owing to the rising demand and unsustainable energy supply, thus having an energy-efficient building is crucial; Buildings designed to provide occupants the necessary comfort while using the minimum required energy. Numerous research has been undertaken to enhance the energy performance of buildings and this thesis will concentrate on two prominent subjects in this field: occupancy prediction and probabilistic load forecasting.

1.1.1 Occupancy Prediction and Energy Usage

According to recent statistics [6], buildings account for 40% of the total energy usage in the world. Occupants' behavior influences significantly that energy usage. In order to reduce energy consumption, using artificial intelligence and smart equipment, it is possible to control HVAC and lighting devices remotely [7, 8]. However, on a daily basis, it is required to tune the HVAC system manually. Recent research works have shown that it is possible to save HVAC energy, by automating its control, in a given building via occupancy detection (i.e. detecting the presence or the absence of occupants inside the building and even inferring their number at a given point of time or within an interval) and prediction [9, 10]. When provided accurate occupancy models, demand-driven control can utilize such information to coordinate real-time HVAC usage, reducing energy use and maintaining indoor thermal comfort in buildings.

1.1.2 Probabilistic Load Forecasting

Since electricity cannot be stored in large quantities and must be supplied as needed, it has become critical for electric power companies to forecast demand ahead of time. Load forecasting is a crucial component in the decision-making processes of the energy industries, whether for fulfilling the demand or for planning and operations [11].

Several studies and methodologies, varying from statistical to machine learning methods, have been established in recent years to improve the process of load forecasting. However, most of these studies are focusing on point forecasting which frequently fails the electric power companies since it ignores the big picture (it neglects information on the uncertainty). Another significant issue is that the majority of solutions are based on recurrent networks or convolutional models. RNNs are notoriously difficult to train due to the gradient vanishing and exploding problem, and despite the development of various variations, such as LSTM and GRU, the concerns remain unresolved [12]. In the case of convolutional-based models, the use of convolutional filters resulted in several modeling restrictions, particularly when modeling long-term and complex relationships in sequence data [13].

This inspired the recent research trend of embracing Transformer-based solutions for time series forecasting. Transformers are models based on a multi-headed attention mechanism that processes the entire sequence of data [14]. Unlike recurrent networks, which analyze their input sequentially, transformers revolve around attention metrics that extend back in time to learn dependencies in the sequence such as seasonality, rendering them suitable to handle longer sequences, overcoming the vanishing gradient problem that impedes Recurrent Networks in the long-term prediction.

1.1.3 Contributions

The contributions of this thesis are as follows:

- ☞ **Weakly Supervised Occupancy Prediction using Training Data Collected via Interactive Learning:**

  The recent prediction studies have shown that the performance of data-driven models depends mainly on three factors [15]: 1) quality of the training data used to learn the models, 2) selection of the input features, and 3) prediction algorithms used for models development. The goal of this study is to develop a non-intrusive, privacy-preserving, and low-cost occupancy prediction unified framework, that takes simultaneously these three factors into account, via some features and information extracted from sensors such as temperature, power consumption, motion detection, humidity, and CO2 data. Different machine/deep learning techniques were considered for prediction. Machine/deep learning has attracted substantial attention from the research community and demonstrated great potential in many energy and building applications. Yet, its use to make accurate occupancy predictions requires large amounts of labeled data. However, labeling occupancy data is cumbersome since it generally needs the involvement of the occupants. In the first study in chapter 2, this problem was tackled by proposing a labeling approach based on an interactive occupancy estimation technique that was previously proposed in [16]. Thus, the prediction framework will be mainly based on a mix of manually and probabilistically labeled training sets which can be viewed as a weakly supervised [17] prediction approach. To the best of the authors' knowledge, it is the first time that a weakly-supervised framework is proposed for occupancy prediction, i.e, the use of small but

high-quality labeled data collected via the interactive learning approach to provide reliable future occupancy predictions. This work [18] was published in Recent Advances in Imaging and Sensing 2022: special issue of Sensors.

☞ **T-DPnet:Transformer-based deep Probabilistic network for load forecasting:**

In chapter 3, a novel parallel encoder-decoder model that takes advantage of the expressiveness of transformer-based encoders to produce probabilistic forecasts is proposed. The model is called T-DPnet, as in a Transformer-based Deep Probabilistic network. In this work, short- and medium-term load forecasting for various datasets is thoroughly investigated in order to incorporate the performance of the proposed framework and demonstrate that it outperforms many existing models. A part of this work was accepted in ITISE 2022 (8th International Conference on Time Series and Forecasting), and an extension was submitted to IEEE Transactions on Industrial Informatics.

1.1.4 Outline

The remainder of the thesis is structured as follows. In chapter 2, the occupancy prediction framework is introduced along with an explanation of interactive learning. Extensive simulations are shown to demonstrate the merits of the proposed prediction framework's performance and to validate the interactive learning-based approach's ability to contribute to the achievement of effective occupancy prediction. Chapter 3, provides the needed background on the key concepts underlying the most used architectures for time-series forecasting, transformers, and their different mechanisms. A comparison to the state-of-the-art models is shown to incorporate the performance of the proposed framework compared to its peers to provide accurate probabilistic forecasts. Finally, chapter 4 provides the concluding remarks and the future research directions.

Chapter 2

Weakly Supervised Occupancy Prediction using Training Data Collected via Interactive Learning

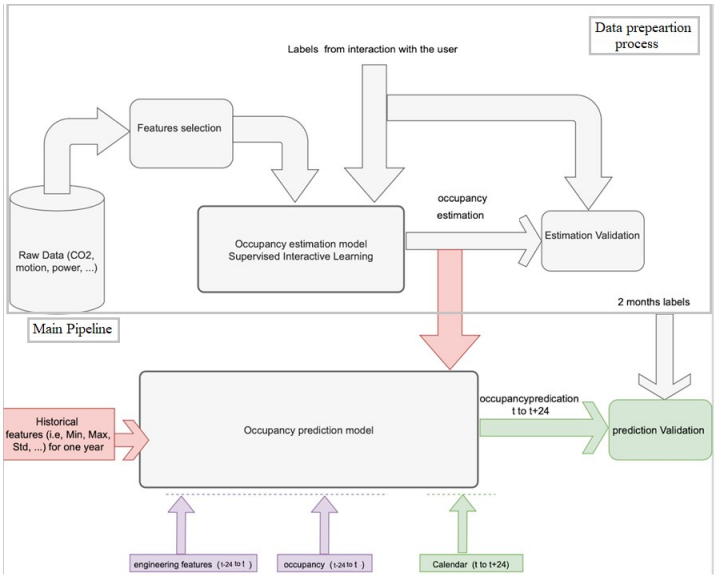
2.1 Related Work

Occupancy prediction is a challenging task due to the stochastic nature (i.e. mainly Markovian) of occupants' behavior [19, 20, 21] and the constraints to obtain training data. In recent years, an increasing number of researchers have used ML methods to create occupancy models. In [22], the behaviour of occupants in a housing complex adjusting thermostat settings and heating system operations was investigated. Then, using machine learning methods such as clustering and decision trees, the occupant behaviour pattern was determined. In [23], the authors developed a statistical machine learning framework, based on random forests, to predict heating load and cooling load of residential buildings with the objective of learning the occupants behaviour patterns. In [24], using a complex sensor network, the authors investigated the relationship between ambient conditions and the number of occupants. CO2 and acoustic metrics have the strongest link with the number of occupants due to the huge number of open offices. Three machine-learning algorithms (Hidden Markov Models, ANN, and SVM) were used to predict the occupancy schedule in a typical day during the test. Some other approaches have been based on the use of sensory data, use of electrical appliances, and water usage, and have deployed mainly machine learning (ML) techniques [25, 26].

2.2 Occupancy Prediction Framework

The purpose of this section is to go over the occupancy prediction framework in depth. The section begins by presenting the overall prediction methodology, followed by a description of the case study under consideration. Finally, the used occupancy estimation approach to generate training data, which was previously proposed and successfully tested in [16], is briefly introduced.

Table 2.1: General occupancy prediction methodology. The following are the two primary processes: 1) the Data preparation process through the generation of specific occupancy labels for existing unlabelled data, and 2) the main pipeline, which leverages the data created by the first step, as well as other extracted features, to estimate occupancy for various window sizes.



2.2.1 Methodology

Due to occupancy's stochastic nature, its short-term prediction for individual rooms remains a challenging task. This study aims to develop a concrete occupancy prediction model by applying machine/deep learning techniques. However, in order to reach the occupancy prediction part, a crucial process is required: the data preparation. Therefore, this study is mainly divided into two steps (As shown in Fig. 2.1): 1) Data preparation process, and 2) Occupancy prediction, the main pipeline.
The first step of occupancy labels preparation is necessary due to the privacy constraints in the case of occupancy prediction problem which makes it challenging compared with load, weather, and solar energy forecasting [27], for instance. To address this issue, first, a feature selection process was carried out on the various raw data acquired by sensor to select the most relevant information. Thereafter, relying on the selected raw data, occupancy data was generated by applying a supervised learning model with an interactive learning approach that depends on a spread rate concept, to validate the quality of generated data, during one year in the office case study[16] (This part is illustrated by the upper part of Fig. 2.1).

Thereafter, once the occupancy estimations are available (provided from the first step), we proceed with the next step: the establishment of the occupancy prediction model (the main pipeline). However, most of the time, occupancy measurements are insufficient to build an accurate and efficient prediction model for at least 24 hours ahead of time; thus, to improve the prediction process, various prior knowledge and engineering features (e.g., the calendar, the office events such as meetings and presentations which most of the time shows an increase in occupancy, the weather and others like the avg/minimum/maximum occupancy values for the past days, etc.) were introduced.

2.2.2 Case study

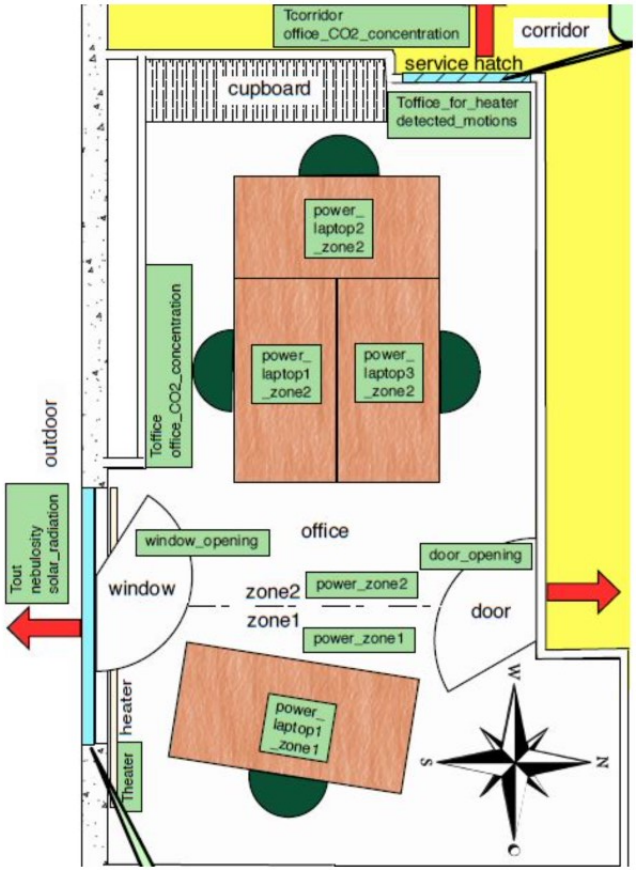


Figure 2.1: Smart office in Ense3, Grenoble Institute of Technology, building.

The testbed setup is at the Grenoble Institute of Technology as illustrated in Fig. 2.1 and described in [28]. It is an office that accommodates a professor and three Ph.D. students. The office has regular visits during the week, with a lot of meetings and presentations. The setup includes different sensing networks that monitor indoor and outdoor temperature, humidity, motion detection, CO2 concentration, and acoustic pressure. It also contains a web application with a consolidated database for continual data retrieval from various sources. Statistical features were extracted from the raw sensory data. Then, the most relevant features were selected employing information gain [29]. The list of features F that will be used in this work is: motion counting, acoustic pressure (from microphone), power consumption, CO2 concentration and door position.

2.2.3 Occupancy estimation and interactive learning

For the case of supervised learning methods in general, and for the estimation of the number of occupants in particular, the problem of the acquisition of accurate target values arises, i.e. the labeling issue. Many of the known and efficient solutions in the literature are based on image processing, information extracted from camera feeds, which is still not acceptable in many locations when it comes to respecting occupants' privacy. Only recently, a few works have proposed occupancy estimation approaches using non-intrusive sensors, with minimal impact on the persons involved. One of the most successful approaches, based on

the concept of interactive learning, has been proposed in [16]. Interactive Learning (IL) is a supervised learning methodology that involves exchanging information with the user to collect a training data set related to a specific context [29]. IL, with a minimal number of interactions, i.e, asks, ensures the collection of good quality data with accurate labels.This approach, proposed in [29], has been successfully applied to estimate the occupancy in office rooms, using different sensors and avoiding the use of cameras [16]. Besides, the concept of interactive learning allows us to evaluate and improve the quality of the database [30].

2.2.3.1 Interactive Learning Methodology

Interactive learning estimates the number of occupants by questioning occupants when relevant, limiting the number of interactions, and maximizing the information usefulness about the actual occupancy [16]. Occupancy estimation algorithms make use of information gathered from occupants as well as common sensors. The interactive learning approach is primarily dependent on interaction methodology to determine when it is necessary to 'ask' the occupants. The *ask* is a question displayed on the screen with its order, date, and time i.e. (Question1, 05/09/2019 15:42:12 How many occupants in last 30 minutes? (0...7)), while in a response area, there are different options to answer, defined according to a minimum and a maximum possible number of occupants with a timeout of 3 hours for each question. Four criteria have been taken into account to determine the interaction time. The first one is the density of the neighborhood, which is defined as the number of existing records (i.e. vectors of sensor features) in the neighborhood of a potential *ask* (i.e. interaction with the occupant). The second criterion is the classifier estimation error in the neighborhood of the potential *ask* which leads to the concept of neighborhood quality that was defined in [30] via a novel concept called spread rate. This methodology is based on the following. If the classifier estimation error is too high for a record, this record is removed from the neighborhood. However, an acceptable estimation error leads to updating the training set with the new record. The third criterion is based on the minimum class weight, which consists of the minimum acceptable number of records for each class. The fourth criteria of spread rate is a global measure of data quality (instead of counting the records, it checks how records are globally distributed). In the IL methodology, the number of occupants is then determined using supervised learning (ex., Random Forest, linear regression, etc.).

2.2.4 Prediction models

The observations about the occupancy are collected at regular time intervals and are time series. The occupancy prediction problem can be stated as follows. Let X^t denote the occupancy during the tth time interval. Given a sequence $\{X^t\}$ of observed occupancy data, $t = 1, 2, \ldots, T$, that include the number of occupants n^t and other information (e.g. calendar, sensor features, etc), the problem is to predict the number of occupants $n^{t+\Delta}$ at time interval $(t + \Delta)$ for some prediction horizon Δ. In this study, the following horizons in terms of hours were considered, $\Delta \in \{1, 12, 24\}$. Predicting the future occupancy will be based on previously collected time series using several well-known approaches from the literature that we shall investigate namely support vector regression (SVR), Multi-Layer Perceptrons (MLP) [31] and Recurrent Neural Network (RNN) [32]. In particular, different RNN variants namely Long short-term memory (LSTM), CNN-LSTM, and Gated Recurrent Unit network (GRU) were considered[32]. In addition, an investigation was carried out to

explore Light Gradient Boosting Machine (LightGBM) which is a gradient boosting open-source framework for gradient boosted machines [33].

2.2.4.1 multilayer perceptron (MLP)

The field of artificial neural networks (ANNs) is often referred to as neural networks or multi-layer perceptrons, the âvanillaâ neural networks. The power of neural networks comes from their ability to deduct the relation between the inputs and outputs, in other words, comprehend the representation in the training data and how to best relate it to the output variable to predict, which is in this case the occupancy.

While a single layer perceptron can only learn linear functions, an MLP is capable to learn both linear and non-linear functions. At each layer, the neurons transform their inputs x_i by calculating a weighted sum z over them (w_i: weights, b: bias), and then this transformation is subjected to a non-linear function ϕ, known as activation function, in order to obtain an intermediate state a. Simply put, an MLP can be defined by the weights between its layers of neurons, the output of which is mostly computed using a nonlinear function. MLPs are popularly known as Universal Function Approximators. They are capable of learning weights that map any input to any output.

$$z = \sum_i w_i \cdot x_i + b \tag{1}$$

$$a = \phi(z) \tag{2}$$

However, this simple architecture faces a lot of challenges, such as the Vanishing and Exploding Gradient. An issue related to the back-propagation algorithm, which is a technique used to update the weights of a neural network by calculating the gradients. In the case of an extremely deep neural network (a large number of hidden layers), the gradient vanishes or bursts as it propagates backward, resulting in disappearing and exploding gradients. Another major defect for this type of architecture is that the latter cannot capture sequential information in the input data which is required for dealing with sequence data which makes it not a great fit for time series forecasting.

2.2.4.2 RNN: LSTM & GRU & bi-LSTM

A looping constraint on the hidden layer of ANN turns to RNN. That is to say that an RNN has an additional looping constraint on the hidden layer than the simple ANN. This addition gives RNN a sense of time context which ensures that sequential information is captured in the input data, hence, overcome one of the ANN major issues. However, this was not sufficient to suppress the other problem, the vanishing gradients. Therefore, new models, such as Long Short Term Memory (LSTM) and Gated recurrent units (GRUs), were introduced. For instance, LSTM relies on a new state called cell state and has a CEC (Constant Error Carousel) which allows the error to propagate back without vanishing. The cell state and its many gates are at the heart of LSTMs. The latter serves as a carrier for relative information to transfer it all the way down the sequence chain. It acts more like the network's "memory", carrying meaningful information throughout the sequence's processing. As a result, knowledge from earlier time steps can be used in later time steps, lessening the impact of short-term memory. As the cell state travels, information is added or deleted from the cell state via gates. The gates are neural networks that determine which

information remains in the cell state. During training, the gates have the capability to learn which information is relevant to keep and which to discard.

As previously stated, the LSTM contains complex dynamics that allow it to easily "memorise" information over a long period of time. The “long term” memory is kept in a vector of memory cells $c_t \in R^n$. The LSTM architecture used in our experiments is given by the following equations [34]:

$$
\begin{aligned}
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i), \\
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f), \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o), \\
g &= \tanh(w_c[h_{t-1}, x_t] + b_c), \\
c_t &= f \odot c_{t-1}^{l} + i \odot g, \\
h_t &= o \odot \tanh(c_t),
\end{aligned}
$$

where i_t represents the input gate, f_t the forget gate, o_t the output gate, σ the sigmoid function, w_x the weight for the respective gate (x) neurons, h_{t-1} the output of the previous LSTM block (at timestamp t-1), x_t the input of current timestamp, b_x the bias for the respective gates (x), g the candidate for cell state at timestamp (t), and c_t the cell state (memory) at timestamp (t).

GRU shares similar characteristics with LSTM. To manage the memorizing process, both algorithms include a gating mechanism. GRU, on the other hand, is less complicated than LSTM and substantially faster to calculate. Later on, other architectures were introduced, for example, bidirectional-LSTM, which is unlike LSTM that only preserves information from previously processed inputs, Bi-LSTM runs the inputs in both ways (from past to future and from future to past), which allow it to preserve information from the past and future.

2.2.4.3 LightGBM

According to [33], LightGBM is a revolutionary gradient boosted decision tree (GBDT) method that arose as a result of the previous implementation of GBDT's lack of efficiency and scalability while dealing with high dimensional features and large data size.

LighGBM is a high-performance gradient boosting framework that utilizes gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB) to enhance computational efficiency without sacrificing accuracy. GOSS is used to split the optimal node by calculating variance gain, whereas EFB can speed up the GBDT training process by grouping numerous exclusive features into fewer dense features [35].

As illustrated in Fig. 2.2, LightGBM develops trees vertically, whilst other algorithms do that horizontally, indicating that LightGBM grows trees leaf-by-leaf, whereas other algorithms grow trees level-by-level, for instance, XGBoost. LightGBM grows the leaf with the greatest delta loss. The Leaf-wise technique minimizes loss more than the Level-wise procedure while extending the same leaf.

2.3 Experimental results

This section is mainly divided into three sections, each of which is strongly influenced by the nature of the estimated data. A discrete occupancy prediction study is presented in

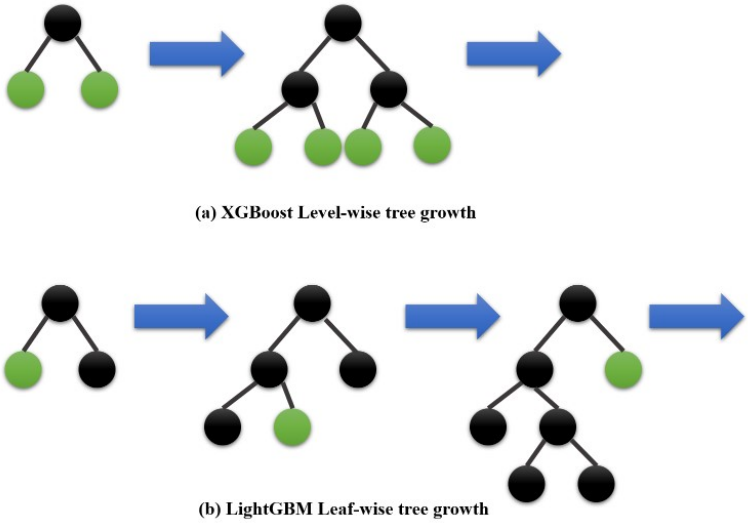


Figure 2.2: XGBoost vs LightGBM tree growth

the first section, where the problem is initially treated with only two classes (0: absence of occupant and 1: the presence of occupant(s)), leading to the second section with three classes (0: absence of occupant, 1: the presence of one occupant, and 2: the presence of more than one occupant). In the second part, the problem's complexity is increased by transforming it into a regression problem rather than a classification problem (continuous data, i.e, the average number of occupants per hour in the office, instead of discrete one). The data was originally collected every 5 minutes, thus we have a 5 min sampled data. Judging that working with hourly sampled data is more efficient (reduced input size), the necessary changes were performed (resampling the data at a 1-hour rate by averaging the sensors reads per hour.)

2.3.1 Additional features extraction

Before diving into the prediction problem, as mentioned before, relying only on occupancy measurements is not enough to feed the prediction model with the necessary information to produce accurate outputs, hence, an investigation of the usage of different features was made to improve the process.

2.3.1.1 DateTime features

Date and time components (month, day of week, and hour) were extracted from the available date-time variable. Such features might seem simple, but in reality, they provide the model with significant information. For instance, as depicted in Fig.2.3(a), it is clear that the highest occupancy levels are spotted at the hour interval between [10 am, 9 pm]. In Fig.2.3(b), the highest occupancy levels were scored on the weekdays as for the weekend, hardly any occupancy was detected. Such features help the prediction model to be more aware of the temporal circumstances.

2.3.1.2 Calendar features

The office calendar is directly connected with the presence of occupants and their activities inside the office which makes calendar features an interesting addition to improve the

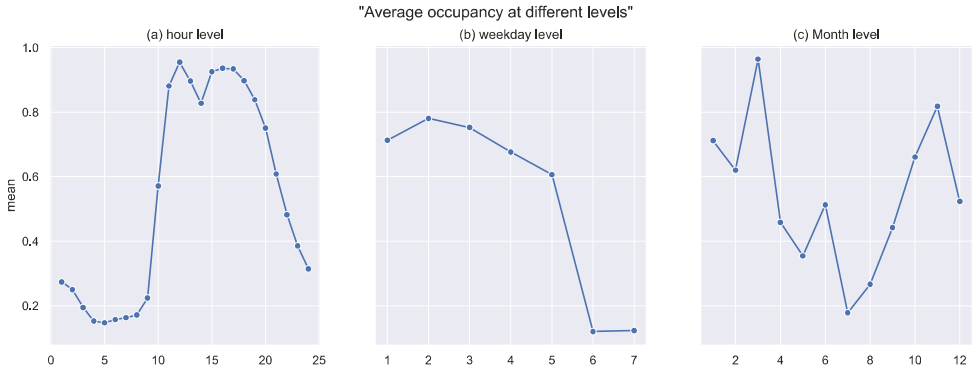


Figure 2.3: Average occupancy at different time levels: (a) hourly level, (b) daily level, and (c) monthly level.

performance of the prediction model. Luckily, such information could be extracted from the Gmail calendar of the main office, which contains all office events (meetings, presentations...), their starting and ending times. Requests were written in a python function to decode the calendar information to be introduced in a data frame that has the same structure as the occupancy data frame to be able to link.

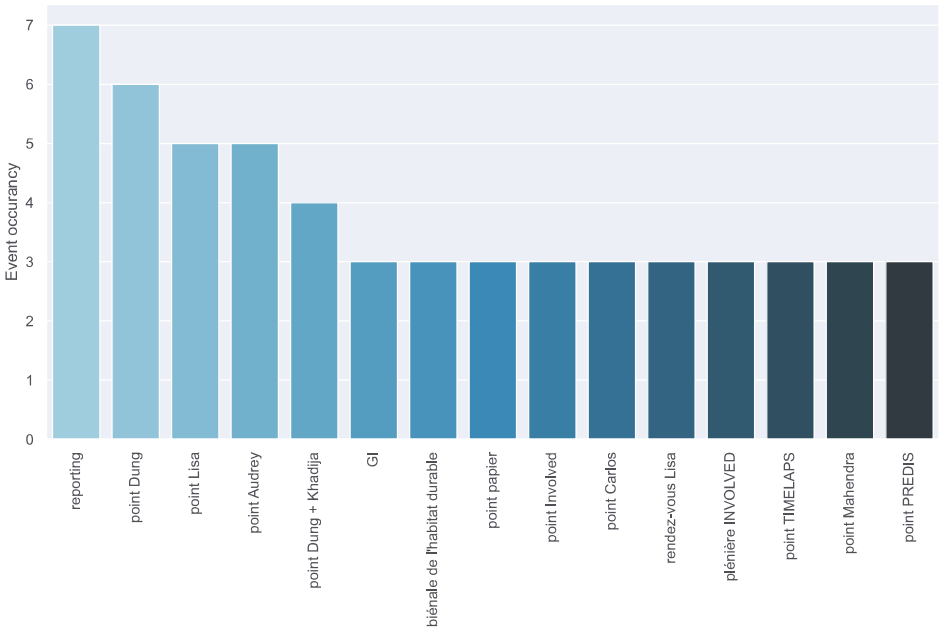


Figure 2.4: The most frequent office events and their count.

Fig.2.4 shows a clear look at the event repetition, where the maximum occurrence is seven times (occurrence) for the event called “reporting" and then most recurrent events are the ones with “point" in their names; these events were coinciding with strong occupancy in the building. After further analysis, it seemed that the calendar events could be divided into 2 categories based on the occupancy levels of the events. Accordingly, in case of the occurrence of an event, the events with “point" or “reporting" were grouped as category referred to as “g2" and all the remaining events were embedded as category “g1", otherwise,

in case there was no event, the sample was reported as category “g0". As shown in Fig.2.5,

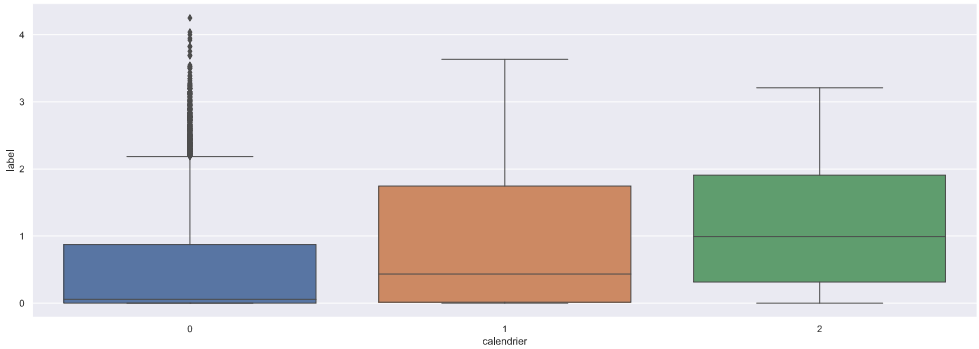


Figure 2.5: Box-plots of the occupancy as a function of calendar categories.

compared to the events in “g2" which match strongly with the occupancy, the “g1" events have an inferior impact on the occupancy in the office. Still, the presence of an event, whether from “g1" or “g2", has a great influence on the occupancy level in the office.

The time and calendar features were both used as lag features (about the historical data) and for the lead features (for the time horizon to be predicted).

2.3.2 Discrete prediction

2.3.2.1 Binary occupancy estimation

The first step in validating the prediction framework is to use an interactive learning approach. Four criteria are used to interact with the user, as detailed in [16]: 1) data quality (spread-rate), 2) density of the neighborhood, 3) minimum class weight, and 4) classifier estimation error.

Six relevant features are used in this experiment as input to the estimation model (1-motion counting, 2-acoustic pressure, 3-power consumption, 4-Co2 concentration, 5- door position, 6-calendar). Considering the case study presented in section 2.2.2, the real-time system was launched for one year. As mentioned previously, for the case of binary prediction, the output of the estimated occupancy data should follow the prediction objective. In other words, the data should consist of just two classes. Different experiments have been done and three methods have been considered and compared for occupancy estimation with interactive learning namely logistic regression, Support Vector Machine (SVM), and Random Forest using just two occupancy levels. According to the results in table 2.2, which shows the quantitative measures for the three models, it is clear that all the three models provided great results, but Random Forest performs the best.

Table 2.2: Estimation results.

ML-algo	precision (%)	recall (%)	f1-score (%)
SVM	95.17	92.99	94.03
Logistic regression	99.25	96.88	98.01
Random Forest	100.0	100.0	100.0

2.3.2.2 Binary occupancy prediction

LightGBM classifier was chosen as a baseline model for this section. The advantage of LightGBM is that even with quick training, it provides accurate results. Hyperparameters tuning was also employed with a 6 fold cross-validation that enabled us to find the best LightGBM model. Coupled with the additional features, the outputs of the estimation model were fed to the prediction model. The preliminary results were not satisfying due to the fact that the data is unbalanced (class 0 covers more than 80% of the total dataset samples) as shown in Fig. 2.6.

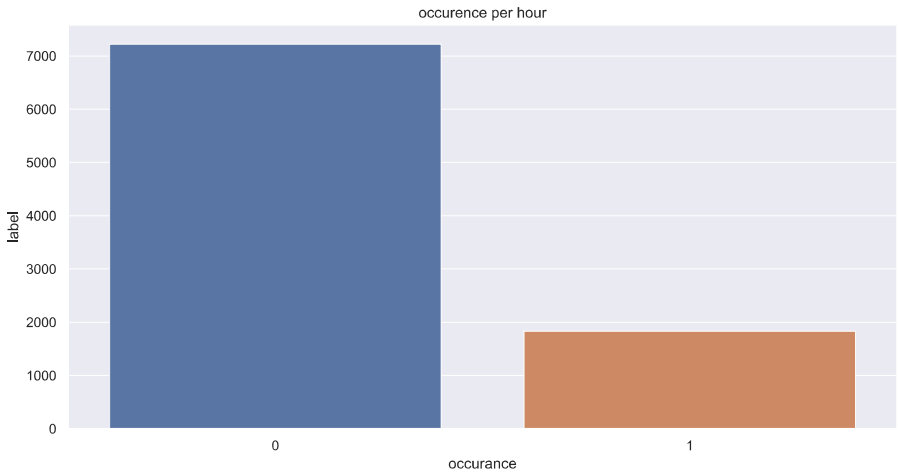


Figure 2.6: Binary Data distribution.

Therefore, more advanced configurations were explored to overcome such challenges. One of the solutions that considerably improved the preliminary results was the use of the class weights. When computing the loss function, classes weights are employed to avoid the model from favoring the major class. If one class dominates the dataset, the model will be biased to learn that class better, because the loss is primarily determined by the model's performance on that dominant class. Weighting involves raising a class contribution to the loss function. As a result, the contribution of the gradient for that specific class will be greater as well. So, instead of using the target values, the gradient and Hessian were multiplied by the weights, which are calculated based on the distribution of the classes.

The final results for the 1/12/24h ahead in time is summarized in table 2.3, where a comparison is carried out between the actual occupancy values with the one provided by the LightGBM model. It is shown that the prediction model learned how to efficiently predict class 0 the dominant class, and also have decent results for the minor class 1 predictions for the different window sizes, especially for the 1h prediction, where the overall precision reached 94%.

The framework performance could be validated by the use case displayed in Fig. 2.7 which shows an 24 hours ahead prediction of one of the weekdays. The model successfully predicted the absence and presence of occupants in the office for almost all 24 hours, except for hour 20, when it incorrectly predicted the presence of the occupant. Following further investigation, it appears that the model has a lacking when it comes to accurately point the time of departure of occupants after lengthy periods of presence which is affecting the metrics shown previously in table 2.3, for the 24h ahead predictions.

Table 2.3: Binary prediction results.

ML-algo	precision (%)	recall (%)	f1-score (%)
1h window			
0: absence of occupants	97.06	97.78	97.42
1: presence of occupants	95.71	94.37	95.04
Overall (macro avg)	96.06	95.42	96.24
12h window			
0: absence of occupants	94.41	93.75	94.08
1: presence of occupants	85.71	87.10	86.40
Overall (macro avg)	90.06	90.42	90.24
24h window			
0: absence of occupants	91.60	92.48	92.04
1: presence of occupants	81.12	79.19	80.14
Overall (macro avg)	86.36	85.84	86.09

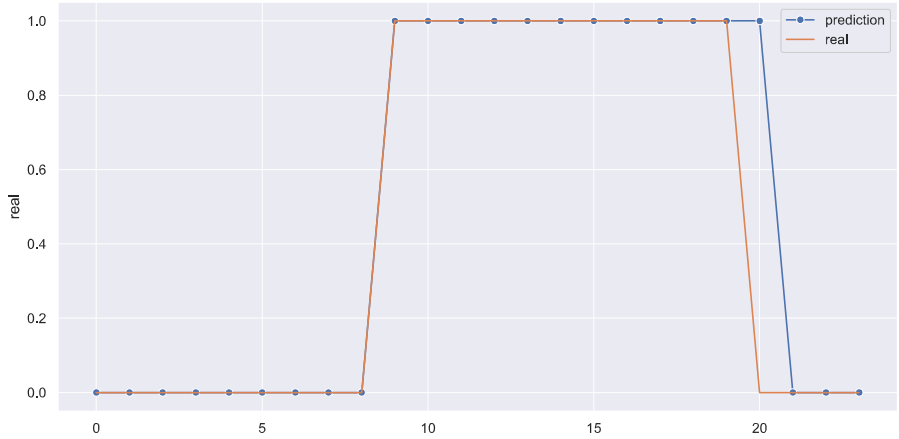


Figure 2.7: The framework prediction versus reality (24h ahead) for the case of binary data (0 or 1).

2.3.2.3 Multi-level occupancy estimation

For the case of multi-level occupancy prediction, the complexity of the problem was increased by adding another level of occupancy. As a result, we have a total of three classes (0: absence of occupant, 1: the presence of one occupant, and 2: the presence of more than one occupant). This change made the estimation part a bit trickier, as evidenced by the results in table 2.4.

Table 2.4: Estimation results.

ML-algo	precision (%)	recall (%)	f1-score (%)
SVM	79.0	76.21	75.16
Logistic regression	94.38	85.19	87.51
Random Forest	95.12	93.4	92.0

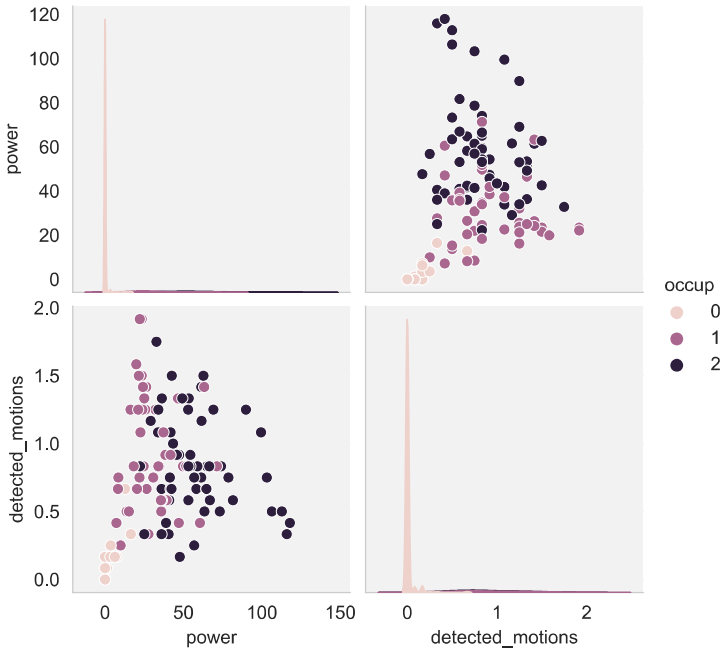


Figure 2.8: Pairplot of power vs detected_motions (avg per hour).

The performances of the three chosen models, in the case of multi-level occupancy, were inferior to theirs while facing binary data. As shown in Fig. 2.8, finding the boundaries to accurately separate the two classes is a bit complicated, especially for classes 1 and 2.

2.3.2.4 Multi-level occupancy prediction

As mentioned previously in the case of binary data, we are facing a problem of unbalanced data where class 0 is dominant, and dividing the occupancy class into two levels, as shown in Fig. 2.9, raised the difficulty bar for the prediction framework.

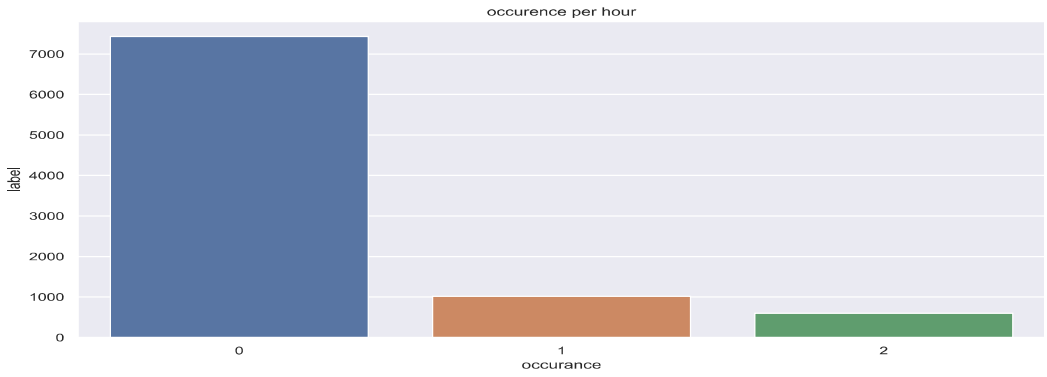


Figure 2.9: Multi-level Data distribution

Table 2.5 summarises the final results for the 1/12/24h ahead in time predictions. It is demonstrated that the prediction model learned how to efficiently predict the dominant class, class 0, but the results for the occupancy levels were unsatisfactory. As shown in Fig. 2.8 and table 2.5, despite effectively learning how to distinguish absence from presence of occupants, the framework is having difficulty specifying the level of occupants, whether

the presence of one or more occupants in the office.

Table 2.5: Multi-level prediction results.

ML-algo	precision (%)	recall (%)	f1-score (%)
1h window			
0: absence of occupants	97.06	93.62	95.31
1: presence of 1 occupant	66.07	86.05	74.75
2: presence of more than 1 occupant	64.29	40.91	50.00
Overall (macro avg)	75.81	73.52	73.35
12h window			
0: absence of occupants	93.60	91.99	92.79
1: presence of 1 occupant	55.09	68.56	61.09
2: presence of more than 1 occupant	57.61	39.85	47.11
Overall (macro avg)	68.76	66.80	67.00
24h window			
0: absence of occupants	93.00	91.61	92.30
1: presence of 1 occupant	42.37	53, 76	47.39
2: presence of more than 1 occupant	52.19	39.53	44.99
Overall (macro avg)	62.52	61.63	61.56

This could be validated by the use case displayed in Fig. 2.10, which shows a 24-hour prediction of one of the weekdays. The model accurately predicted the absence and presence of occupants in the office for almost the entire 24 hours, but it failed to predict the correct number of occupants for a couple of hours. For example, instead of predicting 2, the framework predicted 1 in hours 9, 13, and 16, and in hours 14 and 15, the framework output was 2 instead of 1.

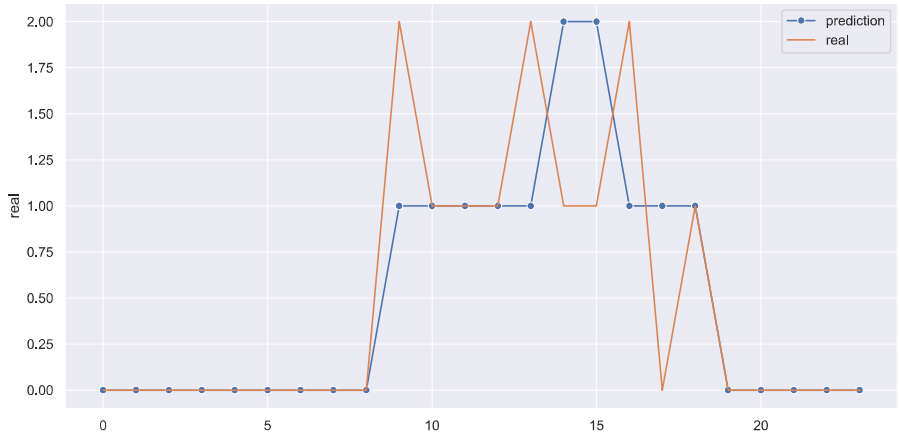


Figure 2.10: The framework predictions for multi-level occupancy versus reality (24h ahead).

2.3.3 Continuous occupancy

In this part, instead of the use of discrete data as in the previous part, continuous data were used. In other words, the faced problem was transformed from a classification one into a regression one.

2.3.3.1 Continuous occupancy estimation

Several experiments were conducted, and three methods for occupancy estimation with interactive learning were considered and compared: linear regression, a gradient boosting regressor, and a Random Forest regressor with five occupancy levels. Furthermore, two evaluation measures were used: 1) Mean Squared Error (MSE), a risk metric corresponding to the expected value of the squared (quadratic) error or loss, and 2) R2 score (R2S), a coefficient of determination (R2) between 0 and 1, with 1 being the best value. Table 2.6 displays the quantitative measures, while Fig.2.11 displays the graphical evaluations of the Random Forest regressor, which outperforms the gradient Boosting regressor and linear regression.

Table 2.6: Estimation results.

ML-algo	MSE	R2S
Linear regression	0.136	0.82
Gradient Boosting	0.084	0.89
Random Forest	0.063	0.91

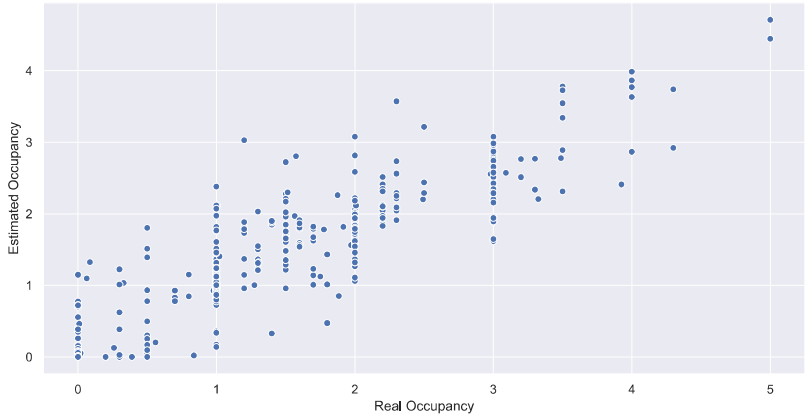


Figure 2.11: Plot of the estimation occupancy and real occupancy when considering Random Forest.

An important parameter in the Random Forest regressor model is the number of trees in the forest. In the presented results this parameter was set by default to 100. Fig. 2.12 depicts the estimation error as a function of the number of trees and shows that changing the number of trees did not significantly reduce the error. When compared to the other two methods, the Random Forest regressor produces the best results. It took 21 requests for training data to create an acceptable estimator. Table 2.7 shows how the 21 questions are distributed across the days using the Random Forest regressor.

Table 2.7: Number of asks each day.

Day	1	2	3	4	5	6	7	8	9	10
Number of *asks*	10	4	0	2	2	0	2	1	0	0

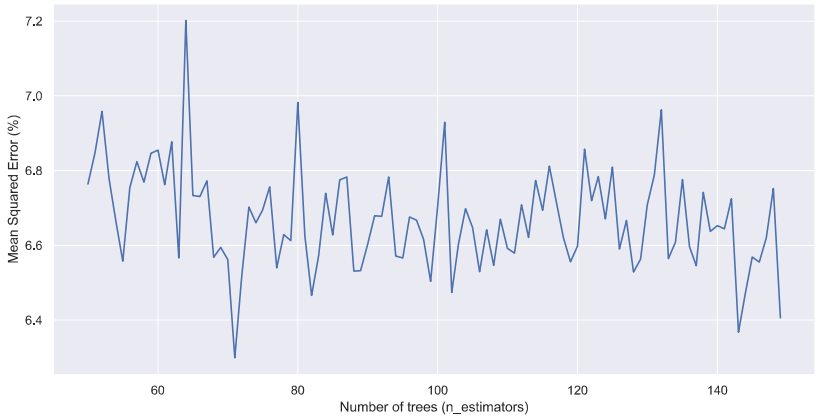


Figure 2.12: Mean squared error as a function of the number of trees.

2.3.3.2 Continuous occupancy prediction.

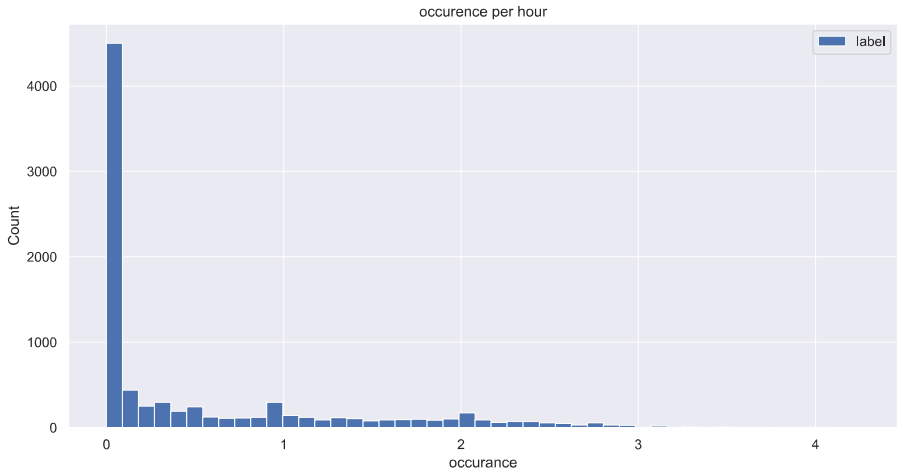


Figure 2.13: Continuous Data distribution (avg number of occupants per hour).

Occupancy estimation using Random Forest regressor and interactive learning provides us with accurately labeled occupancy data to use in the prediction models with an average error of 0.06. In this subsection, different known algorithms were tested to visualize the impact of the additional features on the efficiency of the models. For this study, the occupancy data was used as it is, i.e, continuous data as seen in Fig.2.13. The results for the prediction of the three-time windows (1h/12h/24h ahead) are summarized in table 2.8.

As shown in table 2.8, although the prediction results were close due to the lack in terms of data, LightGBM outperformed the other algorithms, especially for the 12h and 24h ahead predictions. As mentioned previously, LightGBM has the benefit of being able to achieve accuracy with less training time. Furthermore, we decided to investigate the impact of the additional features on the LightGBM performance. As shown in Fig. 2.14, the added features improved the prediction, yet the results were yet not so satisfying for the 12h/24h prediction compared to the ones obtained while using discrete data as expected. After all, the use of discrete data instead of continuous ones would decrease the complexity

Table 2.8: R2 score for different ML-approaches.

ML-regressor	1h	12h	24h
LightGBM	0.831	0.505	0.502
LSTM	0.814	0.360	0.380
CNN_LSTM	0.813	0.367	0.371
Bidirectional_LSTM	0.817	0.363	0.378
GRU	0.805	0.370	0.318
MLP	0.785	0.330	0.370
Regression	0.800	0.400	0.162

of the problem and consequently improve the final results which makes sense since there is less fluctuation in the data.

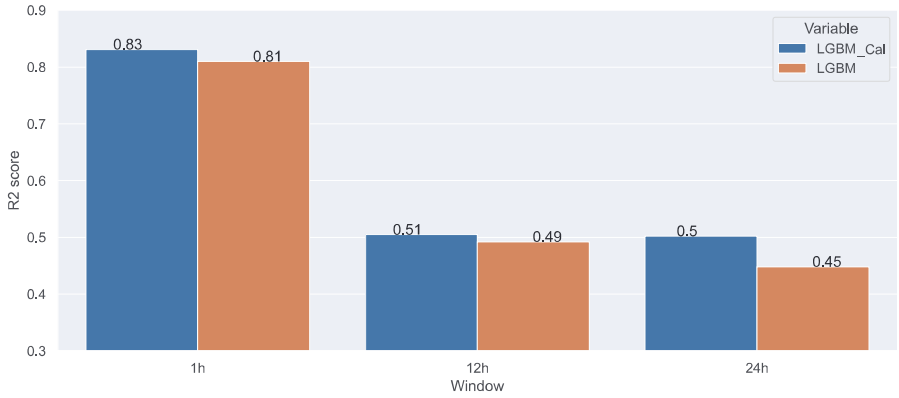


Figure 2.14: Additional features impact on LightGBM predictions.

Chapter 3

T-DPnet: Transformer-based deep Probabilistic network for load forecasting

3.1 Related work

Different studies for load forecasting have been presented in recent years. For instance, the authors in [36] presented several statistical and artificial intelligence techniques for load forecasting, such as support vector machines and linear regression, as well as an analysis to identify the factors that have a significant impact on accuracy, such as weather. The authors in [37] provided an overview of load forecasting, with an emphasis on regression-based models such as regression trees and support vector regression. However, when dealing with large amounts of data, these traditional machine learning techniques seemed to suffer. They are confronted with critical challenges such as scalability [38].

Deep learning algorithms came to the rescue in order to overcome such challenges. Deep Learning algorithms are extremely useful for learning from large amounts of unsupervised data, analyzing data with a logic structure similar to how humans draw conclusions [39]. In [40], the authors evaluated the efficacy of several multi-step techniques on a Multi-Layer Perceptron (MLP), as well as the influence of various factors on bias and variance, such as time series length, and forecasting horizon. The authors in [41], performed a comparison of the prediction performance of various RNNs, such as LSTM, GRU, and NARX, for short-term load forecasting. In [42], the authors introduced an attention model for sequence-to-sequence (Seq2Seq) recurrent neural networks on both univariate and multivariate time series. In [43], The authors developed an encoder-decoder architecture inspired by WaveNet, a model originally built by Google DeepMind for raw audio waveforms that employ dilated causal convolutions and skip-connection to use long-term knowledge to do load demand forecasting 24 hours ahead. In [44], PLCnet, a hybrid deep learning model that combines long short-term memory networks (LSTM) and convolutional neural networks (CNN) models were introduced for short-term load forecasting.

However, the majority of these studies focus on point forecasting, ignoring information on prediction uncertainty, which is critical in time-series forecasting, particularly for demand or load forecasting. In recent years, several studies have been carried out to provide probabilistic forecasting such as the Deep Autoregressive model (DeepAR) that was introduced by Flunkert et al. [45]. DeepAr is a Seq2Seq architecture with an identical

encoder and decoder that estimates the future probability distribution of time series rather than their future values. DeepAR has proven to be one of the most efficient state-of-the-art forecasting models. Under a similar scheme, Ng et al. proposed an MLP-based model to estimate Gaussian parameters with the goal of predicting surgery case duration [46]. Later, Ruofeng et al. proposed Multi-Horizon Quantile Recurrent Forecaster (MQ-RNN) [47]. Relying on its Seq2SeqC architecture and the forking-sequences training scheme that improved the stability and performance of encoder-decoder architecture, MQ-RNN delivers probabilistic multi-step predictions. Rangapuram et al presented deep state space models (DeepState), that combine state space models with deep learning and can retain data efficiency and interpretability while learning the complex patterns from raw data [48]. In 2019, SQF-RNN, a probabilistic framework was proposed by Gasthaus et al. to model conditional quantile functions using monotonic regression splines, allowing for more flexible output distributions [49]. In [50], DeepTCN-Quantile, a probabilistic forecasting approach based on convolutional neural networks (CNN), was proposed by Chen et al. that employs stacked residual blocks based on dilated causal convolutional nets to capture the temporal relationships of the series.

The majority of these probabilistic forecasting frameworks are autoregressive models that use a recursive technique to produce multi-step forecasts and are based on either recurrent neural networks, convolutional neural networks, or both. There are two fundamental issues with such models: The first is the accumulated prediction error caused by training/prediction procedures, while the second is architecture-related issues. To begin, by relying on tactics such as teacher forcing (feeding the model the true value at each successive step in the training phase), minor errors that were not crucial during training quickly get exacerbated over longer sequences during inference. Second, models that rely on recurrent nets are challenging to train due to gradient vanishing and explosion difficulties.

Following its huge success, considered as the new state-of-the-art solution for Natural language processing (NLP) problems, Transformers made their appearance as a solution for time series forecasting. Overcoming the architecture-related issues in recurrent neural nets, a number of Transformers-based solutions have emerged. Wen et al. conducted a thorough review of the most recent transformer schemes for time series modeling, demonstrating transformers' capacity to capture long-range dependencies and interactions, which is a valuable feature for time series modelling [51].

Our method is a parametric approach that estimates the parameters of a probabilistic distribution across predictions, inspired by models such as DeepAr. In addition, our proposed approach is an encoder-decoder architecture that uses the expressiveness of transformer encoders to collect relevant information from historical data in order to make accurate future predictions.

3.2 Background

3.2.1 Sequence models

Traditional machine learning assumes that data points are distributed independently and identically (iid), but when dealing with time-series data, one data item is dependent on those that come before or after it, i.e., the data order is meaningful in such a way that earlier data points or observations provide information about later data points or observations, which inspired researchers to develop sequence models, a more appropriate solution to analyze sequential data [52].

3.2.1.1 Recurrent Neural Network (RNN)

RNN is a type of artificial Neural Network designed for sequential data processing. RNN, unlike standard artificial neural networks (ANN), has internal memory. The design of RNN is inspired by the human brain, in which each cell makes its decision not just on its immediate input but also on the information gained from preceding cells. This enhancement provides RNN with a sense of time context, ensuring that sequential information is captured in the input data [53].

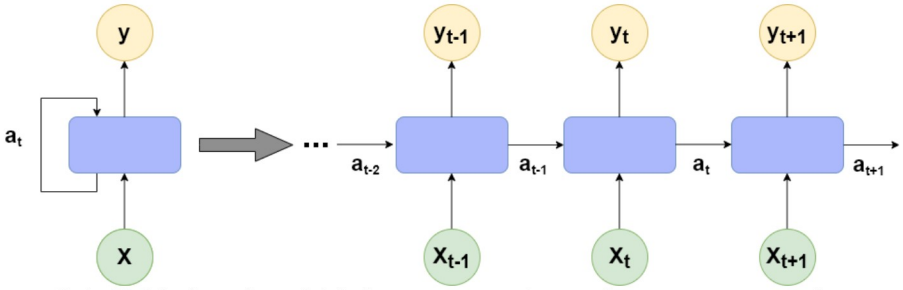


Figure 3.1: Folded and unfolded representations of recurrent neural network.

As illustrated in Fig. 3.1, given x_t, a time-indexed observation vector, for each time-step t, the activation a_t and the output y_t are expressed as follows:

$$a_t = g_1(W_{aa}a_{t-1} + W_{ax}x_t + b_a),$$
$$y_t = g_2(W_{ya}a_t + b_y),$$

where $W_{aa}, W_{ax}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and $g1, g2$ are activation functions.

3.2.1.2 Long Short-Term Memory (LSTM)

RNN tended to suffer while dealing with long sequences as it relied solely on its internal memory to handle sequential data (also known as gradient vanishing and exploding problems). As a result, new models such as Long Short Term Memory (LSTM) were developed. LSTM relies on hidden and gate states to store long-term information [54].

As illustrated in Fig. 3.2, given x_t, LSTM computes a sequence of hidden states h_t, and cell state c_t using a complex architecture which is mainly composed by:

- A forget gate f_t that determines what is retained/lost from the preceding cell state.
- An input gate i_t that determines which sections of the new cell content are written to the cell.
- An output gate o_t determines what parts of cell are output to hidden state.
- A new cell content $\hat{c}_t$.

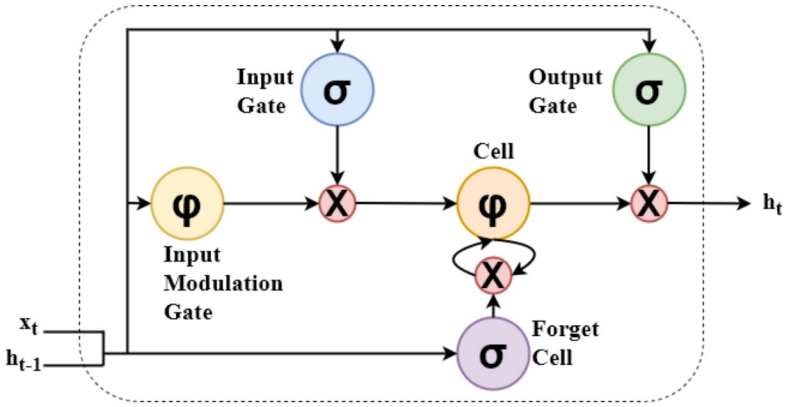


Figure 3.2: Long Short-Term Memory network cell

- A cell state c_t which is responsible on forgetting part of the previous cell state content and inputting some new cell content.
- A hidden state h_t which represents the cell output.

$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f), \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i), \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o), \\
\hat{c}_t &= \tanh(w_c[h_{t-1}, x_t] + b_c), \\
c_t &= f \odot c_{t-1} + i \odot \hat{c}_t, \\
h_t &= o_t \odot \tanh c_t,
\end{aligned}
$$

where σ is the sigmoid function, w_x is the weight for the respective gate (x) neurons, and b_x is the bias for the respective gates (x).

3.2.1.3 Encoder–Decoder

An encoder-decoder architecture is a popular model that was introduced by Cho et al. to solve sequence to sequence problems [55]. As illustrated in Fig. 3.3, the encoder-decoder model, as the name states, mainly consists of two networks: Encoder and Decoder. The first network learns a representation of the input sequence that captures its features or context and outputs a vector which is referred to as the context vector. The context vector is passed to the decoder network, which learns to read it and generate the required outputs.

3.2.2 Transformers

Vaswani et al. were the first to introduce transformers [14]. A deep learning model that employs a self-attention mechanism to weight the inter-relevance of each input data element. One of this model's strong points is its capacity to provide valuable information about distant tokens, i.e. a high capability of learning sequence dependencies such as seasonality in the case of time series.

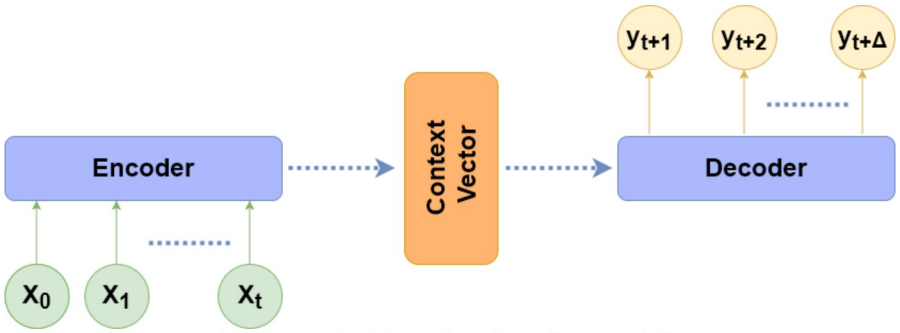


Figure 3.3: Encoder-decoder model.

3.2.2.1 Self-Attention

The transformer's most significant component is the unit of multi-head self-attention mechanism, where Self-attention, also known as intra-attention, is an attention mechanism that relates different positions of a single sequence in order to compute a representation of the same sequence. This mechanism has three significant components:

- Query $\mathbf{Q}$: The query is a representation of the current vector x_t (from the current time-step t) that is used to score against all the other sequence values (using their keys).
- Key $\mathbf{K}$: The key vectors function as labels for all of the values in the segment.
- Value $\mathbf{V}$: Value vectors are the real value representations; once the importance of each value in the sequence is determined, these values are concatenated to represent the current value.

The transformer adopts the scaled dot-product attention which is given by [51]:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}}\right)\mathbf{V}, \tag{3}$$

where $\mathbf{Q} \in \mathcal{R}^{M\times D_k}, \mathbf{K} \in \mathcal{R}^{N\times D_k}, \mathbf{V} \in \mathcal{R}^{N\times D_v}$ and M, N, D_k, and D_v denote the lengths and the dimensions of queries (or keys) and values, respectively.

3.2.2.2 Multi-head attention

Instead of doing the calculations only once, it was proven that the use of the attention mechanism in parallel, i.e., running the attention mechanism multiple times is beneficial to the learning process of the model. Learning from different representations of $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$, helps the model to have a general and a better understanding of the used data. As illustrated in Fig. 3.4, the independent attention outputs are simply concatenated and linearly transformed into the expected dimensions. As a result, the final attention process can be stated as follows:

$$MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Concat(head_1, ..., head_h)\mathbf{W}^O,$$
$$\textbf{where } head_i = Attention(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V),$$

and $\mathbf{W}^O, \mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ are the parameter matrices to be learned.

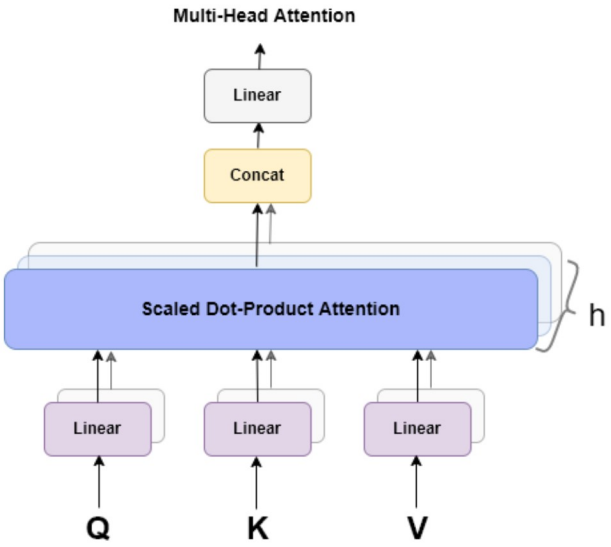


Figure 3.4: Multi-head Attention consists of several attention layers running in parallel.

3.2.2.3 Feed forward and residential network

The vanilla Transformer encoder is composed of multiple identical blocks. Each block, in addition to the multi-head attention layer, contains a fully connected feed-forward network as illustrated in Fig.3.5. Furthermore, each encoder has a residual connection around it and is followed by a layer-normalization step. The residual connections primarily help to alleviate the vanishing gradient problem. The signal is multiplied by the derivative of the activation function during back-propagation. In the case of ReLU, this indicates that the gradient is zero in around half of the situations. If the residual connections were not there, a significant percentage of the training signal would be lost during back-propagation. Because summing is linear with respect to derivatives, each residual block likewise receives a signal that is not impacted by the diminishing gradient. The residual connection summing techniques give a path in the computation graphs that does not lose the gradient.

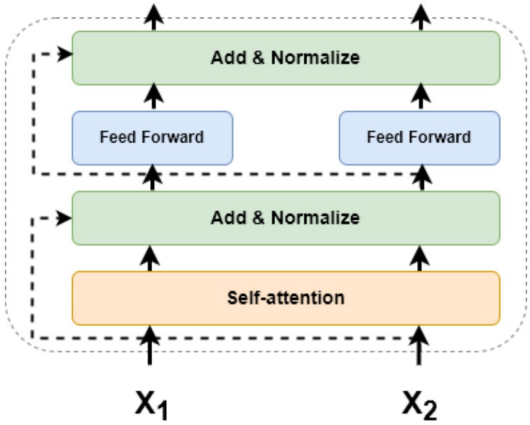


Figure 3.5: Encoder unit architecture.

3.3 Methodology

A general probabilistic forecasting problem can be described as follows: The objective is to model the conditional distribution of the future time series $P(y_{(t+1):(t+\Delta)}|y_{1:t})$, where $y_{1:t} := [y_1, \ldots, y_t]$ refers to the past of the time series, t denotes the length of the historical observations, $y_{t+1:t+\Delta} := [y_t, \ldots, y_{t+\Delta}]$ refers to the future of the time series, and Δ is the length of the forecasting horizon (The following horizons in terms of days were addressed in this work, $\Delta \in \{1, 7, 30\}$, i.e, 1 day, 1 week, and 1 month). However, depending just on past consumption is insufficient for accurate prediction, highlighting the critical importance of including covariates $x_{t+1:t+\Delta}$ that carry meaningful information about the future. The joint distribution of the future incorporating the covariates is expressed as the product of conditional probabilities:

$$P(y_{(t+1):(t+\Delta)}|y_{1:t}) = \prod_{k=1}^{\Delta} p(y_{(t+k)}|y_{1:t}, x_{t+k}) \tag{4}$$

3.3.1 Proposed Architecture

As shown in Fig. 3.6, our model, T-DPnet, is mainly composed of two parallel paths, the backbone and the noise paths with the goal to predict the parameters of the likelihood function, which is a score function used instead of a loss function. For this study, Gaussian likelihood is chosen as it models well the statistical properties of the used data (real-world data). At the core of each path lies a transformer encoder, made up of self-attention and a feed-forward network as explained in the original transformer paper introduced by Vaswani et al. [14], followed by a fully-connected layer that serves as a decoder that outputs the parameters of the probability distribution θ (in our case, it is the mean and the standard deviation). In other words, as in equation 5, we assume that the model distribution consists of a product of likelihood factors parameterized by l'_{t+k} and l''_{t+k}, the latent variables generated by the backbone and noise encoders, respectively.

$$P(y_{(t+1):(t+\Delta)}|y_{1:t}) = \prod_{k=1}^{\Delta} \ell(y_{(t+k)}|\theta(l'_{t+k-1}, l''_{t+k-1})) \tag{5}$$

3.3.2 Processes

3.3.2.1 Likelihood model

The role of the two paths is to generate predictions of the parameters of the hypothetical distribution θ (the mean and standard variation for the case of Gaussian distribution). Once μ and σ are deducted, a maximum likelihood estimation is applied to estimate the corresponding network parameters. The loss function is then created using the negative log-likelihood function as follows:

$$\begin{aligned} L &= -\log \ell(y|\mu, \sigma) \\ &= -\log\left((2\pi\sigma^2)^{\frac{-1}{2}} \exp(\frac{-(y-\mu)^2}{2\sigma^2})\right) \\ &= \frac{1}{2}\log(2\pi) + \log(\sigma) + \frac{(y-\mu)^2}{2\sigma^2} \end{aligned} \tag{6}$$

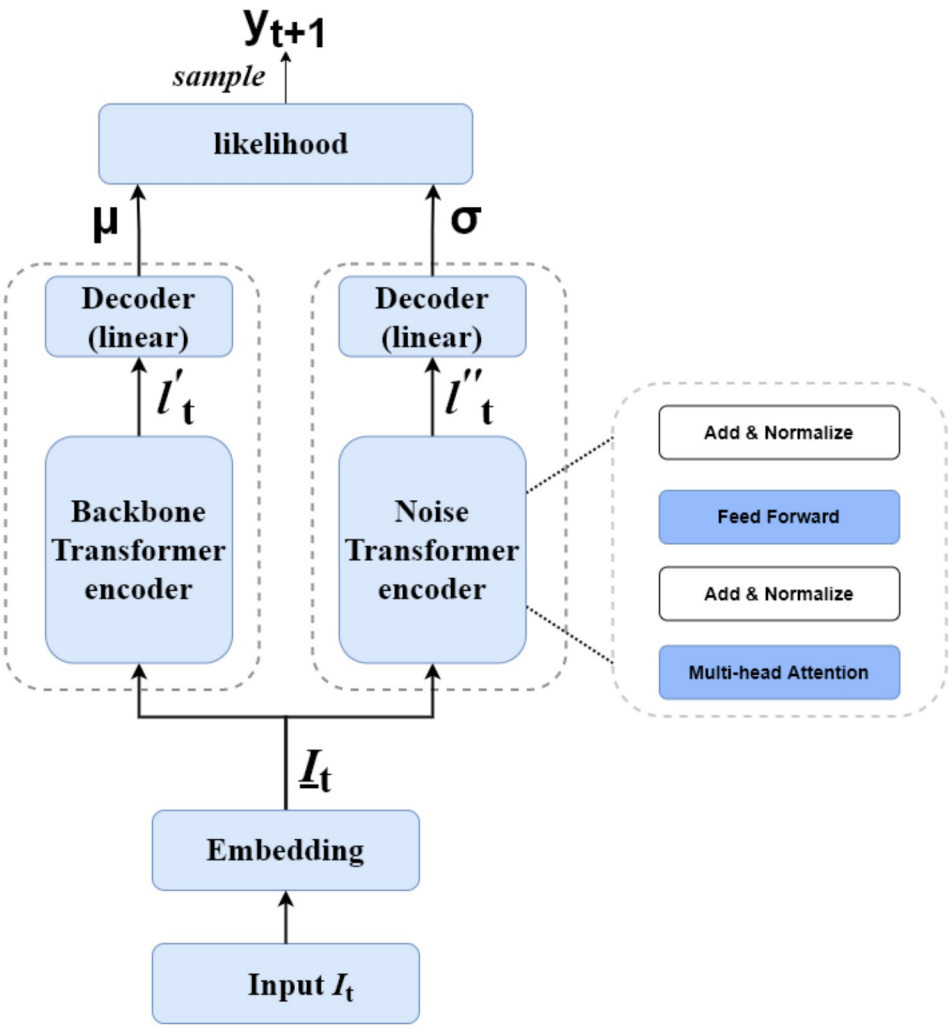


Figure 3.6: Architecture of T-DPnet. An Embedding layer is followed by two paths, each of which contains: an Encoder part with stacked Transformer encoders, each of which consists of a multi-head attention and feed-forward network, and a Decoder part that applies a linear transformation to the encoders' latent variables l'_{t+k} and l''_{t+k} that generates the designed distribution parameters. For prediction, the final emission is generated by drawing a sample $y_{t+1} \sim \ell(y_{(t+1)}|\theta(l'_t, l''_t))$.

3.3.2.2 Input vector

Let I_t be the model input at the t^{th} time interval. As shown in Fig. 3.6, the objective is to predict the energy consumed y_{t+1} based on input I_t. There are two sorts of features in the input vectors: lags and time-dependent features. To avoid data leaking, the former is a list of historical consumption that is determined depending on the forecast horizon size. For example, to predict y_{t+1} for one day, the lag features will start at y_{t-24} as we step back. The latter are time-dependent features related to the $t+1$ slot. It includes the hour, day of the week, and month of the year that characterizes $t+1$, as well as a binary variable that

indicates holidays. For instance, the Δ-hour ahead prediction I_t is built as follows:

$$I_t = [y_{t-\Delta-\Omega}, \ldots, y_{t-\Delta}, h_{t+1}, d_{t+1}, m_{t+1}, hd_{t+1}], \tag{7}$$

where Ω is the length of lags features, $h_{t+1}, d_{t+1}, m_{t+1}, hd_{t+1}$ are the hour of the week, day of the week, month of the year and holiday features that describe time slot $t+1$.

3.3.2.3 Main process

Given I_t, the model input at the t^{th} time interval, our model follows the following process:

$$\begin{aligned}
&Embedding : \underline{I_t} = Embedding(I_t),\\
&Backbonepath : \mu = l_t^{'}\mathbf{A}_{\mu}^{T} + b_{\mu},\\
&\textbf{where} \quad l_t^{'} = BEncoder(\underline{I_t}),\\
&Noisepath : \sigma = \log(1 + \exp(l_t^{''}\mathbf{A}_{\sigma}^{T} + b_{\sigma})),\\
&\textbf{where} \quad l_t^{''} = NEncoder(\underline{I_t}),
\end{aligned}$$

where $\mathbf{A}$ and b are the learnable weight and bias, respectively.

As previously stated, the Gaussian likelihood is parameterized using its mean and standard deviation. The mean is produced by the backbone path, which uses a linear function as the decoder, whereas the standard variation is given by the noise path, which uses a softplus activation on the result of the linear decoder to verify the parameter's positivity.

3.3.2.4 Training vs Prediction

For training, mini-batches are randomly generated from the original time series following the structure mentioned in 3.3.2.2. Those batches are next fed to the model, following the main process mentioned in 3.3.2.3. For each training epoch, once the list of the distribution parameters is generated, losses are computed against the corresponding target. The weights are then updated using gradient descent using back-propagation.

For prediction, depending on the forecasting horizon Δ, inputs with one-period lag are fed to the model. The same process as in the training is followed but with a different final emission. After obtaining the distributions' parameters, instead of calculating loss and performing backpropagation, we simply produce the required quantiles (for example q10, q50, and q90) through repetitive sampling from the resulted distributions.

3.4 Experiments

3.4.1 Real-world datasets

Two real-world datasets are utilized to incorporate the performance of the proposed framework on two different types of data: hourly load data from the power supply company of the city of Johor in Malaysia generated in 2009 and 2010 and hourly load consumption data collected from one of Grenoble Institute of Technology's buildings from 2018 until 2020. The former represents aggregated data, which makes identifying patterns and trends easier, but the latter was taken from a single building (non-aggregated), which increases the difficulty of forecasts. As seen in Fig. 3.7, Grenoble data is significantly noisier than

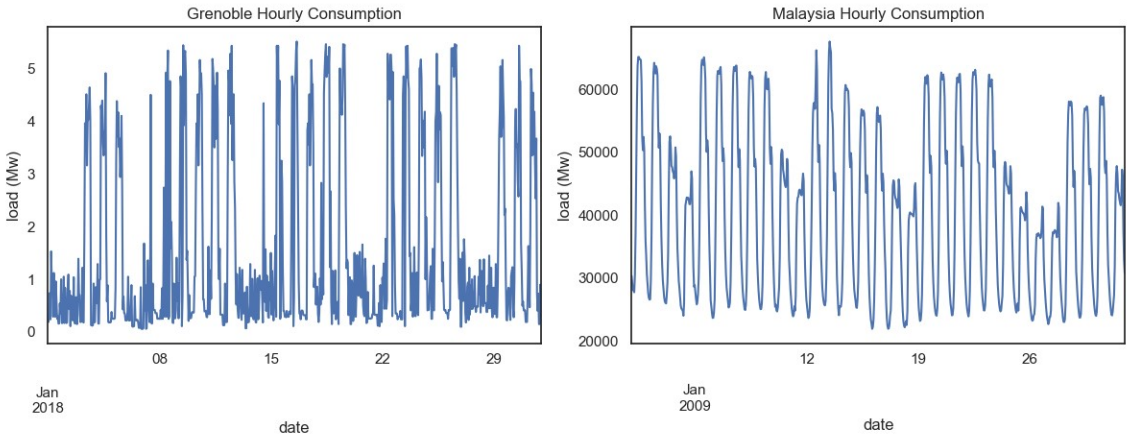


Figure 3.7: Illustration of a sample of Grenoble and Malaysia data (January 2018 and 2009, respectively).

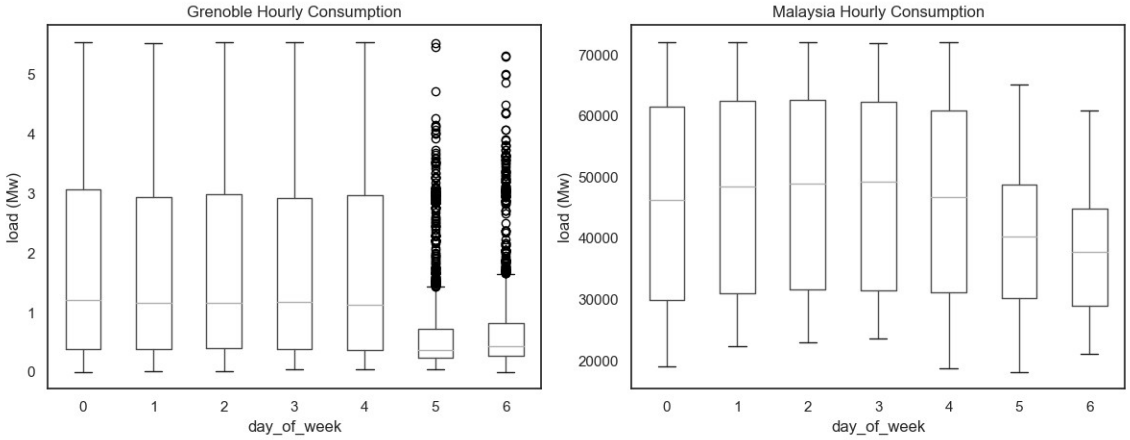


Figure 3.8: Boxplots for Grenoble and Malaysia data hourly electricity Loads vs Day of Week.

Malaysia data, making it more difficult to precisely forecast all spikes when dealing with Grenoble data compared to Malaysia smooth data.

Figs. 3.8 and 3.9 show boxplots for Grenoble and Malaysia data hourly electricity loads for each day of the week and hour of the day, and as anticipated, working days require more electricity than weekends. Peak loads in the Grenoble building occur between 8 a.m. and 4 p.m., whereas peak loads in Malaysia occur between 9 a.m. and 4 p.m. This validates our decision to include time-dependent features, which offer valuable information to the model.

The proposed model performance is discussed across multiple time horizons, including 24-hour, 1-week, and 1-month predictions, versus state-of-the-art models, namely, DeepAr, MQ-RNN, and DeepTCN-Quantile. For both datasets, we trained the models to predict quantiles $q \in \{0.1, 0.5, 0.9\}$.

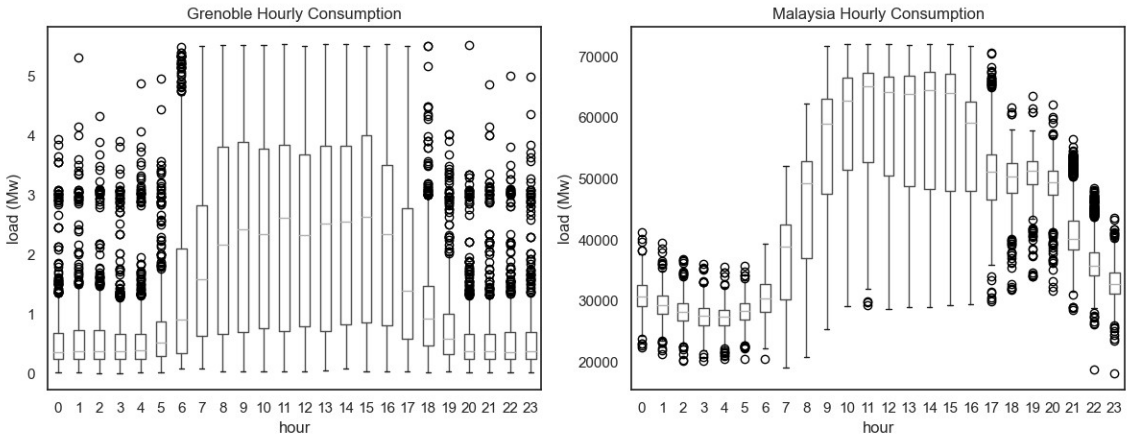


Figure 3.9: Boxplots for Grenoble and Malaysia data hourly electricity Loads vs hour of day.

3.4.2 Evaluation Metrics

As T-DPnet generates probabilistic forecasts, root mean squared error (RMSE), mean absolute error (MAE), and coefficient of determination (R^2) are used to assess the models' one-point forecasting capabilities (P50), coupled with quantile loss (QL) to provide further information about the (P90) quantiles.

$$RMSE = \sqrt{\frac{1}{N}\Sigma_{i=1}^{N}\Big(A_i - F_i\Big)^2} \tag{8}$$

$$MAE = \sum_{i=1}^{N}|A_i - F_i| \tag{9}$$

$$R^2 = 1 - \frac{\Sigma_{i=1}^{N}\Big(A_i - F_i\Big)^2}{\Sigma_{i=1}^{N}\Big(A_i - \hat{A}_i\Big)^2} \tag{10}$$

$$QL = \max\Big(q(F_i - A_i), (q-1)(F_i - A_i)\Big), \tag{11}$$

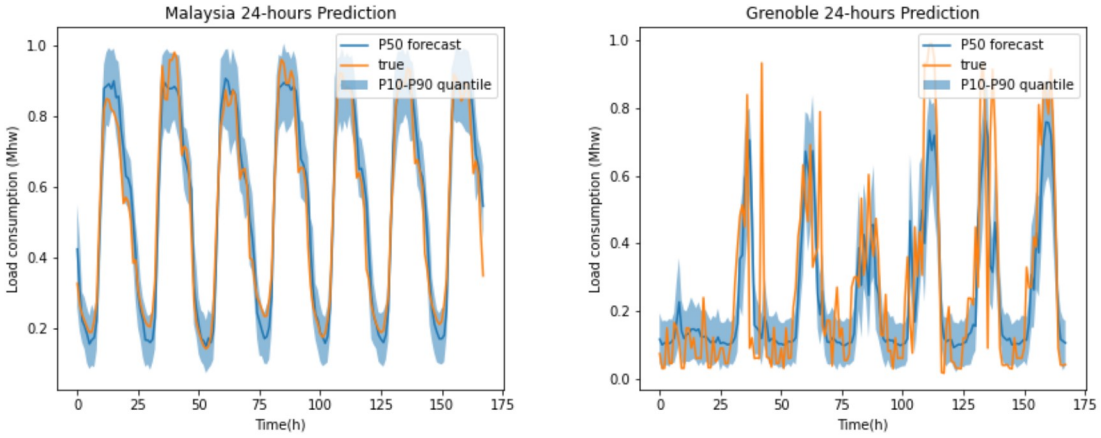
where A_i and F_i refers to actual and forecasted value of the i^{th} point in data, N is the data size, $\hat{A}$ is the average of actual data, and q is the quantile $q \in (0, 1)$. For the case of quantile loss, for a set of predictions, the total loss is the average.

3.4.3 One-day-ahead Forecasting

For the case of a 1-day ahead prediction, a 1-week of historical data is fed to the different models with the goal to predict the next 24 hours. Table 3.1 shows the results for predictions for the next day. T-DPnet outperformed its peers, particularly the state-of-the-art DeepAr. The R^2 was improved from 87.2 percent to 96.2 percent for Malaysian data and from 52.3 percent to 68.2 percent for Grenoble data for 24 hours ahead forecasting. Furthermore, T-DPnet has the lowest error rates in terms of RMSE, MAE, and QLs. Figure 3.10 shows a sample of the predicted test results for both data sets from T-DPnet. The proposed model performs excellently for both data sets.

Table 3.1: Results for 1-day forecasts.

	Malaysia				Grenoble			
	(R^2)	(RMSE)	(MAE)	(Q90/Q50)	(R^2)	(RMSE)	(MAE)	(Q90/Q50)
DeepAr	0.872	0.0963	0.068	0.022/0.034	0.523	0.177	0.127	0.040/0.064
MQ-RNN	0.865	0.109	0.0719	0.0285/0.0409	0.644	0.154	0.112	0.042/0.054
DeepTCN-Quantile	0.890	0.084	0.050	0.024/0.0299	0.667	0.156	0.116	0.036/0.055
T-DPnet	**0.962**	**0.053**	**0.041**	0.010/0.021	**0.682**	**0.149**	**0.103**	0.047/0.0523



(a) 1-day predictions versus actual values for Malaysia data.

(b) 1-day predictions versus actual values for Grenoble data.

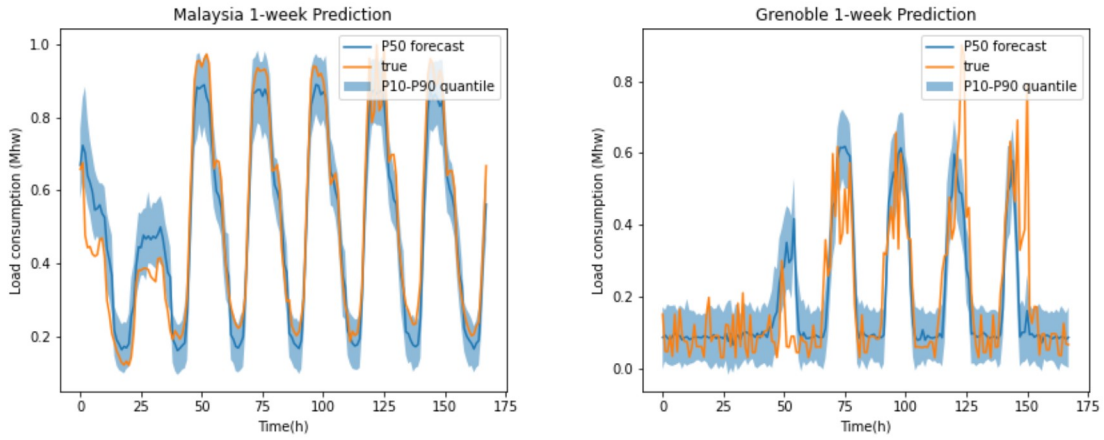
Figure 3.10: T-DPnet 1-day predictions versus actual values for both data sets

3.4.4 One-week-ahead Forecasting

For the case of a 1-week ahead prediction, a 1-week of historical data is fed to the different models with the goal to predict the next 168 hours. Table 3.2 shows the results for predictions for the next day. T-DPnet outperformed its peers while dealing with the Malaysia dataset as the R^2 increased from 87.6 percent to 90 percent versus DeepAr, however, it seems the model is slightly lacking compared to MQ-RNN and DeepTCN while dealing with Grenoble data. T-DPnet results are illustrated in Fig. 3.11.

Table 3.2: Results for 1-week forecasts.

	Malaysia				Grenoble			
	(R^2)	(RMSE)	(MAE)	(Q50/Q90)	(R^2)	(RMSE)	(MAE)	(Q50/Q90)
DeepAr	0.876	0.090	0.065	0.0225/0.032	0.479	0.176	0.126	0.041/0.063
MQ-RNN	0.868	0.093	0.067	0.023/0.035	**0.547**	**0.126**	**0.096**	0.031/0.052
DeepTCN-Quantile	0.890	0.091	0.064	0.022/0.032	0.541	0.200	0.126	0.038/0.058
T-DPnet	**0.900**	**0.081**	**0.059**	0.023/0.03	0.523	0.179	0.119	0.066/0.060



(a) 1-week predictions versus actual values for Malaysia data.

(b) 1-week predictions versus actual values for Grenoble data.

Figure 3.11: T-DPnet 1-week predictions versus actual values for both data sets

3.4.5 One-month-ahead Forecasting

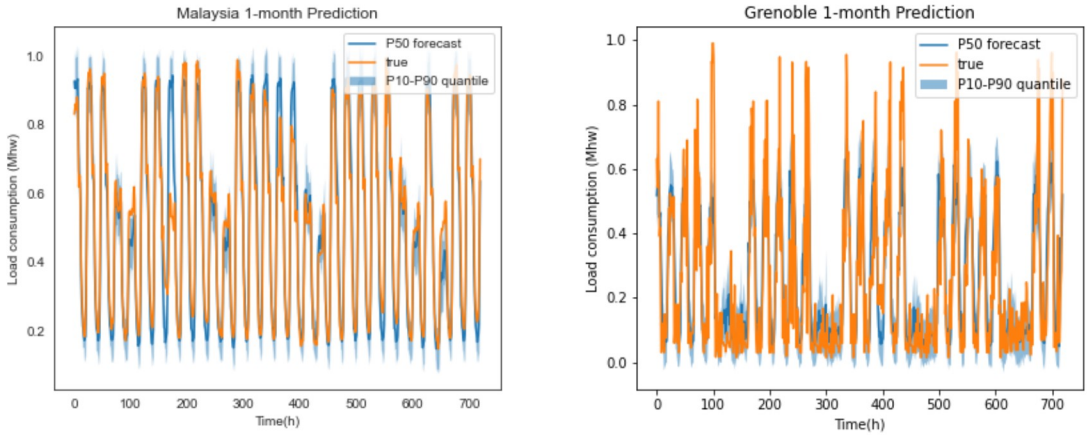
For the case of a 1-month ahead prediction, a 1-month of historical data is fed to the different models with the goal to predict the next 720 hours. As shown in Table 3.3, our model gets better results in long-term forecasting, especially in the Grenoble dataset compared to strong baselines as the R^2 increased from 84.7 percent to 89.7 percent for Malaysian data and from 45.5 percent to 57.2 percent for Grenoble data for 1-month hours ahead forecasting. Fig. 3.12 displays a sample of T-DPnet predicted test results for both data sets. Although the performance of the model decreased compared to shorter sequence forecasting, the results are still considered excellent for load forecasting.

Table 3.3: Results for 1-month forecasts.

	Malaysia				Grenoble			
	(R^2)	(RMSE)	(MAE)	(Q50/Q90)	(R^2)	(RMSE)	(MAE)	(Q50/Q90)
DeepAr	0.847	0.100	0.074	0.0193/0.037	0.455	0.170	0.120	0.040/0.060
MQ-RNN	0.858	0.099	0.069	0.019/0.035	0.450	**0.139**	0.0986	0.36/0.053
DeepTCN-Quantile	0.872	0.095	0.064	0.021/0.031	0.524	0.180	0.124	0.044/0.061
T-DPnet	**0.897**	**0.082**	**0.055**	0.017/0.027	**0.582**	0.170	**0.117**	0.052/0.059

3.4.6 Discussions

In this section, an analysis is performed to determine the performance of T-DPnet in relation to some of the additional categorical features, such as holidays and days of the week, for the case of 24 hours ahead forecast (Malaysia dataset). Firstly, as shown in Fig. 3.13 (a), box plots describe the performance of the model during holidays and no-holidays in terms of R^2 score. The median R^2 score for No-holidays is significantly higher than that of holidays, and there are significant differences in the spreads between the two groups, indicating that the proposed model performs poorly when predicting holiday load consumption. This is due to the fact that the number of observations with holidays in the training data is less

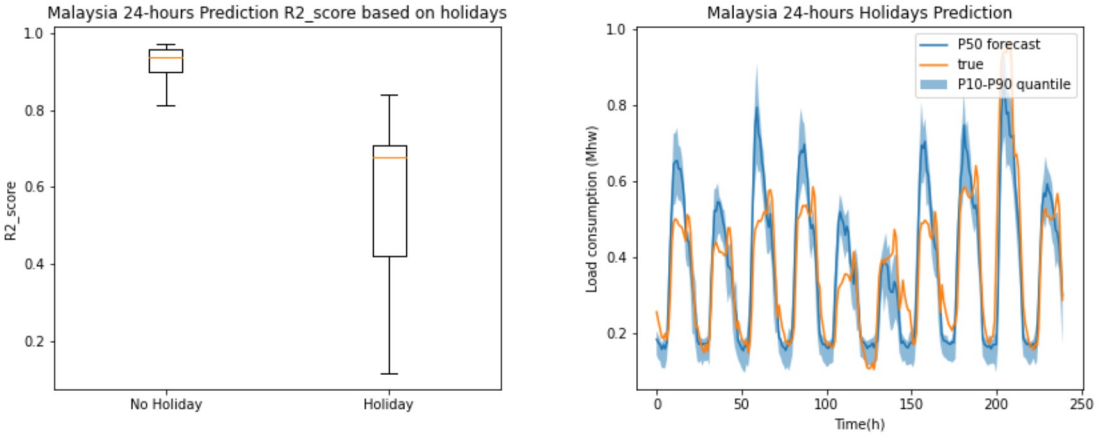


(a) 1-month predictions versus actual values for Malaysia data.

(b) 1-month predictions versus actual values for Grenoble data.

Figure 3.12: T-DPnet 1-month predictions versus actual values for both data sets

than 5 percent of the total number of observations. Nevertheless, in such cases, probabilistic forecasting comes in very handy. Because the uncertainty is higher, probabilistic predictions cover the unexpected, and always attempt to anticipate the worst-case, scenario as shown in Fig.3.13 (b).



(a) R2 score box-plots.

(b) T-DPnet predictions samples for holidays.

Figure 3.13: T-DPnet performance to predict load during holidays.

Still, the performance of the model for such special occasions can be improved by adding features that describe the load for previous events, for example, features representing the mean load consumption of the previous Christmases should greatly improve the load forecast for such event. However, because we were short on data points in this case, we were unable to implement such a solution. Secondly, an illustration of the model performance versus the day of the week is presented in 3.14. The model clearly performs slightly worse on weekends, which is due to the same reason indicated before about the number of observations. During

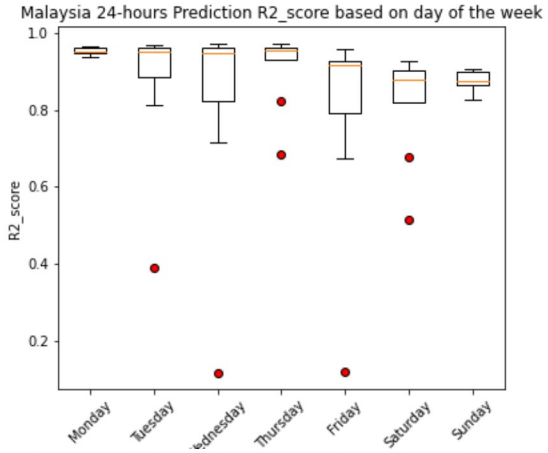


Figure 3.14: T-DPnet performance based on day of the week.

the week, the data follows a daily pattern. In other words, the data from Monday to Friday are comparable in that more than 70 percent of the data represents load consumption during the weekdays. This allows the model to better learn the patterns of these days, resulting in more accurate forecasts. Despite having a high median, $R2$ scores on Tuesday, Wednesday, and Friday are noticeably spread out when compared to the remaining days. Further investigation revealed that the holidays in this data correspond with these precise days, which explains the anomalies.

Chapter 4

Conclusions and Future Work

In this thesis, we addressed two of the most challenging research topics in the field of buildings and energy: occupancy prediction and load forecasting.

The focus of chapter 2 was building an occupancy prediction framework which holds great benefits for building control systems to provide comfort to occupants while saving energy. The proposed solution allows an efficient prediction of the number of occupants in an office relying on the data collected using the interactive approach as well as additional features extracted from the official calendar and event schedules. In the case of the use of binary data, it is shown that the proposed framework, LightGBM-based, can achieve satisfactory performance for different window sizes (1/12/24 hours ahead), reaching a precision of 86% for 24 hours ahead prediction. As for the case of continuous one, extensive simulations show that using non-aggregated data is difficult, particularly when the forecast window size is increased, and that this is attributable to the nature of data (not as smooth as the aggregated one). Recurrent neural network-based models such as LSTM, CNN LSTM, and GRU can produce good results for 1h prediction, but their performance degrades as the window size is increased, with an average accuracy of 37% for 24h prediction. LightGBM marginally suffers from the same dilemma, nevertheless, it achieves the best results among all the discussed models where the accuracy increase from 37% to 50%, which is a good result dealing with this kind of data. Regardless, we believe that apart from the use of information extracted from the office calendar, it would be worthwhile to investigate the association between occupancy prediction and load forecasting. Future research could look into the possibility of using future load prediction as one of the additional features to improve the framework's performance. Another intriguing concept to investigate is using an interactive approach in the occupancy prediction phase, i.e. involving the end-user in the prediction phase as well.

In chapter 3, we introduced a general Transformer-based framework for providing probabilistic predictions for time series data. In contrast to existing sequence-oriented deep learning approaches, our methodology models sequence data using self-attention processes, allowing it to learn complex relationships of varying duration from time series data. Using real-world datasets, we demonstrated that the T-DPnet improves forecast accuracy over the state-of-the-art forecasting methods, the recent RNN, and CNN-based approaches, for short and long-term forecasting. Nevertheless, an important future research direction can be to consider the impact of the addition of other features, such as weather, that should further improve the forecasts

To ensure reproducibility of the results by the research community and a potential

future improvement of the framework by other researchers the complete source codes for both frameworks are provided in the following repositories: https://github.com/OmarBouhamed/Occupancy_pred and https://github.com/OmarBouhamed/T-DPnet.

References

[1] P.-P. Saviotti and A. Pyka, "Innovation, structural change and demand evolution: does demand saturate?," *Journal of Evolutionary Economics*, vol. 27, no. 2, pp. 337–358, 2017.

[2] A. Tealab, "Time series forecasting using artificial neural networks methodologies: A systematic review," *Future Computing and Informatics Journal*, vol. 3, no. 2, pp. 334–340, 2018.

[3] S. Kaushik, A. Choudhury, P. K. Sheron, N. Dasgupta, S. Natarajan, L. A. Pickett, and V. Dutt, "Ai in healthcare: time-series forecasting using statistical, neural, and ensemble architectures," *Frontiers in big data*, vol. 3, p. 4, 2020.

[4] M. Matsumoto and A. Ikeda, "Examination of demand forecasting by time series analysis for auto parts remanufacturing," *Journal of Remanufacturing*, vol. 5, no. 1, pp. 1–20, 2015.

[5] Q. Liu, Z. Li, Y. Ji, L. Martinez, U. H. Zia, A. Javaid, W. Lu, and J. Wang, "Forecasting the seasonality and trend of pulmonary tuberculosis in jiangsu province of china using advanced statistical time-series analyses," *Infection and drug resistance*, vol. 12, p. 2311, 2019.

[6] Z. Chen, Q. Zhu, M. K. Masood, and Y. C. Soh, "Environmental sensors-based occupancy estimation in buildings via ihmm-mlr," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2184–2193, 2017.

[7] X. Lu, F. Feng, Z. Pang, T. Yang, and Z. O?Neill, "Extracting typical occupancy schedules from social media (tossm) and its integration with building energy modeling," *Building Simulation*, vol. 14, pp. 25–41, 2021.

[8] A. Pellegrino, V. R. M. Lo Verso, L. Blaso, A. Acquaviva, E. Patti, and A. Osello, "Lighting control and monitoring for energy efficiency: A case study focused on the interoperability of building management systems," *IEEE Transactions on Industry Applications*, vol. 52, no. 3, pp. 2627–2637, 2016.

[9] D. Sheikh Khan, J. Kolarik, C. Anker Hviid, and P. Weitzmann, "Method for long-term mapping of occupancy patterns in open-plan and single office spaces by using passive-infrared (pir) sensors mounted below desks," *Energy and Buildings*, vol. 230, p. 110534, 2021.

[10] F. Oldewurtel, D. Sturzenegger, and M. Morari, "Importance of occupancy information for building climate control," *Applied Energy*, vol. 101, pp. 521–532, 2013.

[11] T. Hong, P. Pinson, Y. Wang, R. Weron, D. Yang, and H. Zareipour, "Energy forecasting: A review and outlook," *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 376–388, 2020.

[12] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, p. III–1310–III–1318, JMLR.org, 2013.

[13] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, pp. 162–169, 02 2017.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[15] T. Liu, Z. Tan, C. Xu, H. Chen, and Z. Li, "Study on deep reinforcement learning techniques for building energy consumption forecasting," *Energy and Buildings*, vol. 208, p. 109675, 2020.

[16] M. Amayri, S. Ploix, N. Bouguila, and F. Wurtz, "Estimating occupancy using interactive learning with a sensor environment: Real-time experiments," *IEEE Access*, vol. 7, pp. 53932–53944, 2019.

[17] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, pp. 44–53, 08 2017.

[18] O. Bouhamed, M. Amayri, and N. Bouguila, "Weakly supervised occupancy prediction using training data collected via interactive learning," *Sensors*, vol. 22, no. 9, 2022.

[19] D. Yan, X. Feng, Y. Jin, and C. Wang, "The evaluation of stochastic occupant behavior models from an application-oriented perspective: Using the lighting behavior model as a case study," *Energy and Buildings*, vol. 176, pp. 151–162, 2018.

[20] J. Page, D. Robinson, N. Morel, and J.-L. Scartezzini, "A generalised stochastic model for the simulation of occupant presence," *Energy and Buildings*, vol. 40, no. 2, pp. 83–98, 2008.

[21] Z. Chen, J. Xu, and Y. C. Soh, "Modeling regular occupancy in commercial buildings using stochastic models," *Energy and Buildings*, vol. 103, pp. 216–223, 2015.

[22] X. Ren, D. Yan, and T. Hong, "Data mining of space heating system performance in affordable housing," *Building and Environment*, vol. 89, 2 2015.

[23] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," *Energy and Buildings*, vol. 49, pp. 560–567, 2012.

[24] A. Yezioro, B. Dong, and F. Leite, "An applied artificial intelligence approach towards assessing building performance simulation tools," *Energy and Buildings*, vol. 40, pp. 612–620, 12 2008.

[25] V. L. Erickson and A. E. Cerpa, "Occupancy based demand response hvac control strategy," in *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, (New York, NY, USA), p. 7â12, Association for Computing Machinery, 2010.

[26] S. Lee, Y. Chon, Y. Kim, R. Ha, and H. Cha, "Occupancy prediction algorithms for thermostat control systems using mobile devices," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1332–1340, 2013.

[27] C. Fan, M. Chen, R. Tang, and J. Wang, "A novel deep generative modeling-based data augmentation strategy for improving short-term building energy predictions," *Building Simulation*, vol. 15, pp. 197–211, 2022.

[28] M. Amayri, S. Ploix, P. Reignier, and S. Bandyopadhyay, "Towards Interactive Learning for Occupancy Estimation," in *ICAI'16 - International Conference on Artificial Intelligence (as part of WORLDCOMP'16 - World Congress in Computer Science, Computer Engineering and Applied Computing)*, (Las Vegas, United States), July 2016.

[29] M. Amayri, S. Ploix, P. Reignier, and B. Sanghamitra, "Towards interactive learning for occupancy estimation," in *Proceedings of ICAI'16 - The 18th International Conference on Artificial Intelligence*, pp. 1–9, 2016.

[30] M. Amayri, S. Ploix, N. Bouguila, and F. Wurtz, "Database quality assessment for interactive learning: Application to occupancy estimation," *Energy and Buildings*, vol. 209, p. 109578, 2020.

[31] R. Collobert and S. Bengio, "Links between perceptrons, mlps and svms," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, (New York, NY, USA), p. 23, Association for Computing Machinery, 2004.

[32] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[33] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[34] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 6645–6649, IEEE, 2013.

[35] C. Chen, Q. Zhang, Q. Ma, and B. Yu, "Lightgbm-ppi: Predicting protein-protein interactions through lightgbm with multi-information fusion," *Chemometrics and Intelligent Laboratory Systems*, vol. 191, pp. 54–64, 2019.

[36] V. Gupta, "An overview of different types of load forecasting methods and the factors affecting the load forecasting," *International Journal for Research in Applied Science and Engineering Technology*, vol. V, pp. 729–733, 04 2017.

[37] B. Yildiz, J. Bilbao, and A. Sproul, "A review and analysis of regression and machine learning models on commercial building electricity load forecasting," *Renewable and Sustainable Energy Reviews*, vol. 73, pp. 1104–1122, 2017.

[38] M. M. Ahsan, M. A. P. Mahmud, P. K. Saha, K. D. Gupta, and Z. Siddique, "Effect of data scaling methods on machine learning algorithms and model performance," *Technologies*, vol. 9, no. 3, 2021.

[39] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, "An introductory review of deep learning for prediction models with big data," *Frontiers in Artificial Intelligence*, vol. 3, 2020.

[40] S. B. Taieb and A. F. Atiya, "A bias and variance analysis for multistep-ahead time series forecasting," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 1, pp. 62–76, 2016.

[41] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, "An overview and comparative analysis of recurrent neural networks for short term load forecasting," *CoRR*, vol. abs/1705.04378, 2017.

[42] Y. G. Cinar, H. Mirisaee, P. Goswami, É. Gaussier, A. Aït-Bachir, and V. V. Strijov, "Time series forecasting using rnns: an extended attention mechanism to model periods and handle missing values," *CoRR*, vol. abs/1703.10089, 2017.

[43] F. Dorado Rueda, J. Durán Suárez, and A. del Real Torres, "Short-term load forecasting using encoder-decoder wavenet: Application to the french grid," *Energies*, vol. 14, no. 9, 2021.

[44] B. Farsi, M. Amayri, N. Bouguila, and U. Eicker, "On short-term load forecasting using machine learning techniques and a novel parallel deep lstm-cnn approach," *IEEE Access*, vol. 9, pp. 31191–31212, 2021.

[45] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[46] N. H. Ng, R. A. Gabriel, J. McAuley, C. Elkan, and Z. C. Lipton, "Predicting surgery duration with neural heteroscedastic regression," in *Proceedings of the 2nd Machine Learning for Healthcare Conference* (F. Doshi-Velez, J. Fackler, D. Kale, R. Ranganath, B. Wallace, and J. Wiens, eds.), vol. 68 of *Proceedings of Machine Learning Research*, pp. 100–111, PMLR, 18–19 Aug 2017.

[47] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A multi-horizon quantile recurrent forecaster," *arXiv preprint arXiv:1711.11053*, 2017.

[48] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[49] J. Gasthaus, K. Benidis, Y. Wang, S. S. Rangapuram, D. Salinas, V. Flunkert, and T. Januschowski, "Probabilistic forecasting with spline quantile function rnns," in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics* (K. Chaudhuri and M. Sugiyama, eds.), vol. 89 of *Proceedings of Machine Learning Research*, pp. 1901–1910, PMLR, 16–18 Apr 2019.

[50] Y. Lin, I. Koprinska, and M. Rana, "Temporal convolutional attention neural networks for time series forecasting," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021.

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

[52] H. Yousuf, M. Gaid, S. Salloum, and K. Shaalan, "A systematic review on sequence to sequence neural network and its models," *International Journal of Electrical and Computer Engineering*, vol. 11, 10 2020.

[53] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[54] O. Bouhamed, M. Amayri, and N. Bouguila, "Weakly supervised occupancy prediction using training data collected via interactive learning," *Sensors*, vol. 22, no. 9, 2022.

[55] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1724–1734, Association for Computational Linguistics, Oct. 2014.