Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Henrik Larsson Hestnes

Machine Learning-Enabled Predictive Modeling of Building Performance for Electricity Optimization

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
June 2023

NTNU
Norwegian University of Science and Technology

Henrik Larsson Hestnes

Machine Learning-Enabled Predictive Modeling of Building Performance for Electricity Optimization

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

With the rapid advancement of machine learning, predictive modeling has emerged as an attractive avenue for optimizing energy usage in the context of building performance. This thesis delves into the exploration of different modeling paradigms for forecasting the indoor temperatures of buildings, with the ultimate goal of optimizing electricity consumption.

The thesis investigates three prominent modeling paradigms: Physics-Based Modeling (PBM), Data-Driven Modeling (DDM), and Hybrid Analysis and Modeling (HAM). PBMs leverage mathematical equations derived from physical laws to forecast the dynamics of a system, whereas DDMs exploit patterns in historical data to make forecasts. Finally, HAMs aim to incorporate the best of both worlds, combining physics-based models with data-driven components.

The study conducts a comprehensive comparison, assessing each paradigm's ability to accurately forecast indoor temperatures for electricity optimization purposes. From the comparative analysis, DDMs, particularly the Long Short-Term Memory (LSTM) model, outperform the other models regarding accuracy and reliability.

However, despite the success of the LSTM, this study identifies the model's inherent "black box" nature as a weakness. Furthermore, the model's low interpretability can hinder its trustworthiness and its subsequent real-world application, marking an area for further development.

On the contrary, although the HAM model implemented in this study does not provide satisfactory results, it showcases significant potential in enhancing the interpretability and trustworthiness of accurate forecasting models. Thus, the thesis underlines the need for a more extensive exploration of the HAM paradigm.

Finally, the study emphasizes incorporating indoor temperature forecasting models into optimization algorithms as a pivotal direction for a more sustainable future. Such a model would need to handle control inputs like radiators, ventilation systems, and fireplaces, enabling it to forecast indoor temperatures based on different control input sequences. Implementing a confidence measure to assess the reliability of the forecast is also suggested.

In conclusion, the thesis sheds light on the substantial potential of predictive modeling, particularly the DDM and HAM paradigms, for optimizing electricity consumption and reducing electricity costs toward a sustainable future. Finally, the thesis highlights areas that need further exploration and development.

Sammendrag

Med den raske fremgangen i maskinlæring har prediktiv modellering dukket opp som en attraktiv metode for å optimalisere energibruk, også innen bygningssektoren. Denne avhandlingen utforsker ulike modelleringsparadigmer for å forutsi temperaturen i bygninger, med det ultimate målet om å optimalisere energibruken.

Avhandlingen undersøker tre fremtredende modelleringsparadigmer: fysikkbasert modellering (PBM), datadreven modellering (DDM) og hybrid analyse og modellering (HAM). PBMer benytter matematiske ligninger avledet fra fysiske lover for å forutsi systemets dynamikk, mens DDMer utnytter mønstre i historiske data for å forutsi fremtidig utvikling. HAMer tar sikte på å kombinere det beste fra begge verdener ved å slå sammen fysikkbaserte modeller med datadrevne komponenter.

En omfattende sammenligning blir utført for å vurdere hvert paradigmes evne til å nøyaktig forutsi innendørstemperaturer med tanke på å optimalisere strømforbruket. Fra den komparative analysen utmerker DDMene seg, spesielt Long Short-Term Memory (LSTM) modellen, som overgår de andre modellene i form av nøyaktighet og pålitelighet.

Til tross for LSTMens suksess, blir modellens iboende "black box"-natur identifisert som en svakhet. Modellens lave tolkbarhet kan hindre dens troverdighet og dens påfølgende anvendelse i den virkelige verden, og markerer et område for videre utvikling.

På en annen side viser HAMen implementert i denne studien, til tross for utilfredsstillende resultater, betydelig potensial for å forbedre tolkbarheten og påliteligheten til slike prediktive modeller. Derfor understreker avhandlingen nødvendigheten av mer omfattende utforskning av HAM-paradigmet.

Til slutt fremhever studien inkorporeringen av prediktive bygningsmodeller i en optimaliseringsalgoritme som en viktig retning for en mer bærekraftig fremtid. Modellen må da endres slik at den kan håndtere kontrollinnganger som radiatorer, ventilasjonssystemer og peisovner, noe som gir den evnen til å forutsi temperaturer basert på forskjellige sekvenser av kontrollinnganger. Implementeringen av et konfidensmål for å vurdere påliteligheten til prediksjonen er også foreslått.

For å konkludere kaster avhandlingen lys over det betydelige potensialet til prediktiv modellering, spesielt paradigmene DDM og HAM, for å optimalisere energiforbruk og redusere strømkostnad for en bærekraftig fremtid. Til slutt fremhever avhandlingen områdene som trenger videre utforskning og utvikling.

Preface

This thesis marks the culmination of my Master of Science degree in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU), which concludes 18 years of education. The journey that led to the writing of this thesis has been both challenging and rewarding. I have genuinely appreciated the opportunity to work on a real and opportune problem with real-life time-series data, thereby expanding my understanding of the intricacies within predictive modeling of indoor temperatures, which potentially can transform the way electricity is used in the building sector. The goal was ambitious – to evaluate various modeling paradigms and their efficacy in forecasting the indoor temperatures of a building, with the ultimate aim of optimizing energy usage.

This work would not have been possible without the support and guidance of my advisor, Professor Adil Rasheed, from the Department of Engineering Cybernetics. His invaluable insights and constructive feedback paved the way for a comprehensive and insightful exploration of this complex topic. I am also grateful to the department for providing me with the resources necessary to carry out this research.

Finally, I would like to express my deepest gratitude to my family, especially my mom and dad, for their unwavering support. Their encouragement helped me persist, and their faith in my capabilities was one of the driving forces that motivated me throughout these years of education.

In this thesis, I have endeavored to comprehensively compare different modeling paradigms, hoping to add value to the ongoing conversation about predictive modeling of indoor temperatures and its potential applications for energy optimization. Furthermore, I believe this research will contribute toward the development of indoor temperature forecasting models for energy optimization, paving the way for a more sustainable future.

Writing this thesis has been an enlightening adventure that has enriched my understanding and honed my research skills. I hope readers find this work informative and inspiring as they navigate the complex yet captivating world of predictive modeling for indoor temperature forecasting.

For the sake of completeness, this thesis includes the following material from the specialization project leading up to this thesis[1]. Additionally, chapter 1 is inspired by the introduction from the same project.

- Sections 2.1 and 2.2
- Figure 4.5 and table 4.4

Trondheim, May 2023
Henrik Larsson Hestnes

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

PBM	Physics-Based Model(ing)
DDM	Data-Driven Model(ing)
HAM	Hybrid Analysis and Modeling
LSTM	Long Short-Term Memory
CoSTA	Corrective Source Term Approach
FF	Feed-Forward
FFNN	Feed-Forward Neural Network
NLP	Natural Language Processing
RNN	Recurrent Neural Network
HVAC	Heating, Ventilation, and Air Conditioning
PCA	Principal Component Analysis
GPU	Graphics Processing Unit
ESP-r	Environmental Systems Performance - Research
MSE	Mean Square Error
MAE	Mean Absolute Error
GHI	Global Horizontal Irradiance
NCCS	Norwegian Centre for Climate Services
RC	Resistor–Capacitor

1. Introduction

The ongoing energy crisis plaguing Europe has prompted a stronger emphasis on sustainability and reducing electricity consumption. The building sector is a major contributor to both of these issues, accounting for 30% of the global energy consumption and 27% of the global CO_2 emissions from the energy sector[2]. Providing thermal comfort and acceptable indoor air quality alone accounts for a considerable 35% of the energy consumed by buildings[3].

To tackle this issue, it is imperative to improve buildings' energy efficiency while maintaining the desired level of thermal comfort. One appropriate solution is the utilization of accurate predictive models of indoor temperature evolution as a cornerstone for optimizing the electricity used for providing thermal comfort. Even minor enhancements in energy efficiency in the building sector would lead to significant global energy savings and environmental benefits.

The material in this chapter is inspired by the introduction from the specialization project leading up to this thesis[1].

1.1. Background and Motivation

The energy crisis currently affecting Europe has led to a surge in the energy price in 2022 and 2023[4]. This surge has been further exacerbated by the intentional reduction of gas supply from Russia, which is closely linked to electricity prices in the European Union's internal market. The recent conflict between Russia and Ukraine has highlighted vulnerabilities in our energy systems, including supply, consumption, and pricing[4]. Both individuals and companies are facing significant costs associated with maintaining indoor climate control, such as heating, cooling, and ventilation.

These challenges are not limited to Europe, and the increasing global demand for energy has led to similar issues in other regions[5, 6]. Thus, a comprehensive and sustainable approach is necessary to address these problems and ensure our energy systems' long-term stability and security.

The traditional way to reduce the energy consumption of buildings is to improve their insulation through design or refurbishment, which reduces heating or cooling loss. This reduction can be accomplished by installing double- or triple-glazed windows and using better thermal insulation in outer walls or roofs, among other techniques[7]. However, many modern countries, such as Norway, already have strict regulations regarding the insulation of buildings, and the room for improvement may be minimal. This traditional approach also comes with a significant expense.

A more innovative approach involves equipping buildings with digital tools that allow

for automatic adjustments of heating, lighting, and other systems based on the number of people present at any given time. This approach requires real-time monitoring, analysis, and action, utilizing objective tools to measure and calculate when and how to act[7]. Due to the steady price decrease of sensors and computing power over the past few decades, this approach has the potential to be significantly cheaper and, due to recent breakthroughs in the machine learning paradigm, also more efficient than the traditional method.

The optimization of energy usage for climate control necessitates the development of advanced control systems, with accurate and reliable indoor temperature forecasting models providing the basis for the control algorithms. Such models can forecast the impact of different control inputs on the temperature, allowing the control system to make informed decisions. By combining an indoor temperature forecasting model with a control algorithm that optimizes the given cost function, which comprises energy prices or energy consumption, significant energy cost or consumption savings for individuals and companies can be achieved. For example, Nest Labs reported that their smart learning thermostat resulted in an average of 10%-12% savings on heating usage and 15% savings on cooling usage for homes with central air conditioning[8].

Pairing such a control algorithm with an accurate indoor temperature forecasting model of the building could lead to even more significant energy savings. Electricity prices are inherently fluctuating, and an examination of the electricity price in eastern Norway on the 23rd of May 2023 proved a decrease of 95% between the highest and lowest electricity price during the day[9]. Furthermore, with the increasing penetration of wind energy and its inherent volatile nature, which can cause fluctuations in the cost of electricity[10, 11], the ability to optimize the timing of electricity usage based on these price fluctuations becomes even more valuable.

At present, indoor temperature forecasting models can be categorized into two types: Physics-Based Models (PBM) and Data-Driven Models (DDM). PBMs are constructed based on fundamental physical principles and reasoning, making them interpretable and applicable to a wide range of problems, including those involving different buildings with similar physics. These models can also handle extrapolation based on fundamental physics[12, 13]. However, they require numerous assumptions regarding factors such as the heat capacity and heat transfer coefficient of walls, the heat emitted by occupants, radiators, and fireplaces, and the opening and closing of doors and windows. Any slight inaccuracy in these assumptions can accumulate and offset the forecast. These models are also prone to numerical instability and are often computationally expensive.

On the other hand, DDMs employ historical data and experiences to learn the system's dynamics and make forecasts. Once trained, they are highly stable within their interpolation range and computationally efficient for forecasting. However, when extrapolating, the models are unable to confine the errors or uncertainties, which may lead to unreliable forecasts. Moreover, these models are influenced by bias in the data, and their performance is limited by the quality of the data on which they are trained. Finally, such models are generally challenging to interpret due to their "black box" nature[12].

A third emerging paradigm capable of modeling indoor temperatures is Hybrid Analysis and Modeling (HAM). The HAM paradigm seeks to combine the PBM and DDM

approach to leverage and retain their individual strengths while minimizing their weaknesses[12]. By doing so, the PBM can be designed as a more general PBM applicable to numerous buildings, while the DDM can be trained to resemble the unique dynamics of the building's temperature. Moreover, this approach is capable of handling the cold-start issue often seen in DDMs. This problem arises during the initial period when the system has not yet gathered sufficient data to accurately model the DDM. In the early stages, the model can lean more heavily on the PBM. As the DDM starts to learn the unique dynamics of the building's temperature, it gradually provides increasingly accurate forecasts over time.

The specialization project leading up to this thesis investigated how a state-of-the-art PBM developed in the simulation software ESP-r can be utilized for indoor temperature forecasting[1]. This thesis will therefore look into how DDM and HAM can be utilized for the same purpose. Two different DDMs, Long Short-Term Memory (LSTM) and Transformer, will be developed and examined. Additionally, the novel yet promising HAM architecture CoSTA will be explored.

The physical asset for this study is a smart house located by Jonsvatnet, Trondheim, Norway. This asset already has an existing digital twin, and one of the aims of this work is to expand upon this digital twin by forecasting the indoor temperatures of the asset. The asset has over a year of outdoor and indoor temperature measurements in seven rooms, which will serve as the dataset for the DDMs and HAM. The measurements will also serve as the ground truth for evaluating and assessing the forecasts produced by the models.

1.2. Contribution, Research Objectives and Research Questions

1.2.1. Contribution

The main contribution of this work is to develop and examine different architectures for indoor temperature forecasting. The work will thoroughly analyze, examine and discuss the performance of two DDM architectures, namely LSTM and Transformer, and one HAM architecture, CoSTA, and compare their performance with a state-of-the-art PBM.

Even though the concept of Transformers is well-known and widely used in the field of Natural Language Processing (NLP), its application outside the NLP field has been minimal[14]. This work intends to contribute to filling this knowledge gap by examining its performance on time-series data from a physical system. LSTMs are already known to be effective in modeling various physical dynamics[14], and this work intends to expand on this knowledge by examining its performance in forecasting the indoor temperatures of a building.

This work will also look into the paradigm of HAM, more specifically the CoSTA architecture, and examine how this architecture performs in forecasting the indoor temperatures of a physical asset. To the best of the author's knowledge, this has not been done before, and the work intends to address some of the existing knowledge gaps in this paradigm. This architecture has several advantages over purely data-driven architectures and has the potential to outperform DDMs on the proposed problem. Earlier studies have shown great promise for CoSTA[12, 13].

1.2.2. Objectives

To guide this study, a set of research objectives are formulated. In addition, research questions leading to the research objective are presented.

Primary Objective: Develop, examine, and juxtapose indoor temperature forecasts from different architectures and assess their applicability as cornerstones for electricity optimization.

Secondary Objectives:

- Develop a thorough understanding of PBM, DDM, and HAM for indoor temperature forecasting.
- Assess the strengths and weaknesses of PBM, DDM, and HAM for indoor temperature forecasting.
- Evaluate and assess the suitability of PBM, DDM, and HAM as cornerstones for electricity optimization.

1.2.3. Research Questions

The following research questions govern the research produced by this study.

- Which PBM, DDM, or HAM architecture forecasts the evolution of the indoor temperatures of a building most accurately?
- How reliable are the forecasts from the most accurate model?
- Are the forecasts from the most accurate model applicable as a cornerstone for electricity optimization?

1.3. Structure of the Thesis

The structure of this thesis is organized into six chapters that cover different aspects of the research.

- Chapter 1 introduces the study and presents its motivation. It provides an overview of the research topic and the problem being addressed. In addition, it presents the objectives and research questions of the study, which are used throughout the thesis.
- Chapter 2 presents the background theory that this work is built upon, which mainly has been taken from existing literature after the proper citation. First, dynamic systems modeling in the context of PBM, DDM, and HAM is introduced before fundamental theory and specific approaches for the different modeling paradigms are presented.
- Chapter 3 introduces the dataset utilized in this study and the preprocessing steps performed in order to obtain a more suitable dataset.
- Chapter 4 presents the proposed methods for indoor temperature forecasting. The chapter is split up into the three modeling paradigms, and further data preprocessing and specific implementation details for each paradigm are presented.

- Chapter 5 presents the forecasts from the proposed architectures and discusses the results. After that, their applicability as cornerstones for electricity optimization is discussed. Finally, any lessons learned are presented and discussed.
- The final chapter, chapter 6, concludes on the research questions proposed in section 1.2.3 and briefly discusses the implications and potential applications of the findings. Finally, further work is presented and briefly discussed.

2. Theory

This chapter presents the relevant theory for the work done in this thesis. First, the concept of dynamic systems modeling and three of its modeling paradigms will be introduced, namely PBM, DDM, and HAM. After that, the three paradigms will be further delved into, and specific approaches for indoor temperature forecasting from each paradigm will be presented.

In this chapter, the material in sections 2.1 and 2.2 is primarily from the specialization project leading up to this thesis[1] and is included for completeness.

2.1. Dynamic Systems Modeling

Dynamic systems modeling models complex systems over time, used to describe and forecast interactions between multiple system components. It involves representing the relationships among the various components of a system to understand how the system behaves and evolves. One key feature of dynamic systems modeling is using linear and non-linear equations to describe the system's behavior. Non-linear equations are equations that, unlike linear equations, do not have a constant rate of change. Non-linear equations are often used to model systems that exhibit complex behavior, such as oscillations or chaotic behavior[15]. Another key feature of dynamic systems modeling is the use of feedback loops. Feedback loops are mechanisms by which the outputs of a system are fed back into the system, thus influencing its future behavior. For example, in a simple thermostat, the temperature inside a building is measured and used to control the heating system. When the temperature falls below a certain threshold, the heating system is turned on, and when it rises above a certain threshold, it is turned off. This feedback loop helps to maintain a stable temperature inside the building[16].

Dynamic systems modeling has been used in various fields, including biology, economics, and engineering. For example, in biology, dynamic systems models have been used to study the behavior of populations, ecosystems, and the spread of diseases. In economics, dynamic systems models have been used to study the behavior of financial markets and the effects of economic policies. Finally, in engineering, dynamic systems models have been used to design control systems for complex systems such as aircraft and power plants[15, 17].

Overall, dynamic systems modeling is a powerful tool for understanding and forecasting the behavior of complex systems over time. By representing the interactions and feedback loops among the components of a system, dynamic systems models can provide insight into the causes of complex behavior and help to design effective control strategies.

2.1.1. Physics-Based Modeling - PBM

PBM is a method of modeling complex systems using the principles of physics and reasoning to describe the behavior of the system[13]. This approach is commonly used in

engineering to simulate the behavior of physical systems, such as the motion of objects, the flow of fluids, and the interactions of forces. By utilizing the laws of physics to describe the behavior of a system, PBMs can, with some accuracy, forecast the motion of objects, the flow of fluids, and the effects of forces on a system. This approach has a wide range of applications, including modeling blood flow, heat transfer, mass transfer, and flow around wind turbines, to name a few[12].

One advantage of PBM is its ability to provide interpretable, generalizable, and trustworthy simulations of large and complex systems[12]. By using the principles of physics to describe the interactions among the various components of a system, PBMs can simulate the behavior or evolution of systems over time. This can be useful in fields such as civil engineering, where the behavior of large structures such as bridges and buildings needs to be simulated and analyzed, or any other physical system subject to be simulated and analyzed. PBM is a powerful tool for understanding and forecasting the behavior of complex physical systems[12]. By using the laws of physics to describe the behavior of a system, PBMs can provide insight into the underlying mechanisms that drive a system and help to design effective control strategies.

However, there are some limitations to PBM. One of the main challenges of this type of modeling is the need to derive governing equations for the system being studied. These equations are often not fully descriptive of the system, as unknown physics may not be accounted for. Furthermore, deriving these equations almost always involves making assumptions, which can result in a loss of accuracy or completeness. Additionally, these equations may be difficult to solve analytically, requiring numerical techniques, which can also introduce errors or inaccuracies. Another limitation of PBM is that it can be computationally demanding, making it less efficient than other modeling paradigms[13]. Additionally, PBMs are typically not able to adapt to real-time measurements of physical states, which can limit their usefulness in some situations. Despite these limitations, PBM can still be a valuable tool for understanding and forecasting physical systems, particularly when the underlying physical principles are well understood, and the model is carefully constructed and validated[12].

2.1.2. Data-Driven Modeling - DDM

DDM is a modeling approach that uses data to build and train models. DDMs prosper on the premise that large quantities of data manifest both known and unknown physics and seek to learn the complete physics of a system by utilizing large quantities of data to identify complex patterns and relationships[13]. This approach differs from traditional physics-based approaches, which typically rely on assumptions and theoretical foundations to develop models, as described in section 2.1.1. DDM has become increasingly popular in recent years due to several advances in the development of powerful machine learning algorithms and the increased accessibility of large quantities of data. These algorithms can automatically identify and learn complex patterns and relationships in data and can be used to build forecasting models that can be used for a wide range of applications[12].

One example of DDM is the use of Feed-Forward Neural Networks (FFNN) or Recurrent Neural Networks (RNN) for predictive modeling. In this type of modeling, data is used to train a model to make predictions about future events or outcomes. For instance, a model might be trained on data about historical temperatures to make predictions or,

more specifically, forecasts about future temperatures. Another example of DDM is the use of clustering algorithms to identify patterns and relationships in data. Clustering algorithms can be used to group data into clusters based on similar characteristics and can be helpful in various applications, such as market segmentation and anomaly detection.

Overall, DDM is a powerful tool for uncovering patterns and relationships in vast amounts of data and can be used to build models that make accurate forecasts and facilitate better decision-making. One of the significant advantages of DDMs compared to PBMs is their ability to automatically identify and learn complex patterns and relationships in data, allowing them to model known and unknown underlying physics without any explicit information about the underlying physics. The models are also highly computationally efficient once trained, and if trained properly, can achieve high accuracy even for very challenging problems[12]. DDMs can also easily be retrained with new data, enabling them to adapt and model even newly introduced physics in an online and self-adapting manner[18].

However, DDMs also have some limitations. One of the main drawbacks of DDMs is that they often take the form of an uninterpretable "black box". It is not possible to determine the bounds on errors or uncertainties in DDMs, and these models do generally not perform well when extrapolating beyond the range of data they were trained on, which limits their usefulness in high-stakes and safety-critical applications[19]. No sound theory exists for model stability analysis, which further dilutes the trustworthiness of the model[13]. Additionally, DDMs are only as good as the data on which they were trained, meaning that any biases or weaknesses in the data will be reflected in the model, which sets requirements for the quality of the data[19].

2.1.3. Hybrid Analysis and Modeling - HAM

HAM is an emerging modeling approach combining different techniques to better understand and forecast the behavior of complex physical systems. This approach can be useful in various fields, including engineering and physics, and seeks to combine PBM and DDM to leverage and retain their strengths while minimizing their weaknesses. The HAM paradigm, therefore, contains powerful tools for modeling physical systems[13].

For example, DDMs often lack generalizability and struggle to account for intricate physical phenomena, and their interpretability is generally low. Yet, these are areas where PBM shines, boasting both generalizability and interpretability. On the flip side, PBMs can be computationally expensive and may require detailed input data, which can be challenging to obtain[12]. Additionally, PBMs are often based on significant assumptions, which will offset the model if the assumptions do not reflect the actual system. These shortcomings of PBM are where DDMs excel. HAM overcomes these limitations by integrating the knowledge gained from both approaches and playing on their individual strengths, leading to more accurate and reliable forecasts.

A promising example of a hybrid model is the Corrective Source Term Approach (CoSTA). CoSTA utilizes a PBM in conjunction with a DDM responsible for modeling the source term or error in the underlying equations of the PBM. More specifically, CoSTA utilizes the DDM to predict the source term of the governing equations of the PBM and then adds this source term to the discretized PBM. By doing so, the model can account for

unresolved physics in the PBM, which the PBM could not model independently. CoSTA seeks to use the PBM to the greatest extent possible to retain interpretability and achieve excellent accuracy. The most recent results for CoSTA have been promising[12].

The following table sums up the strengths and weaknesses of the three modeling paradigms with the indoor temperature forecasting task in mind. Note that this table not serves as a general comparison and highly depends on the specific situation or applications.

Characteristic	**PBM**	**DDM**	**HAM**
Generalizability	Good	Bad	Good
Interpretability	Good	Bad	Good
Computational Efficiency	Bad	Good	Neutral
Accuracy	Bad	Good	Good
Requires Domain Knowledge	Yes	No	Yes

Table 2.1.: Strengths and weaknesses of PBM, DDM, and HAM

2.2. Physics-Based Building Temperature Modeling

Physics-based building temperature modeling is a method of modeling the temperature of a building by taking into account the physical properties of the construction, such as thermal conductivity and specific heat capacity, as well as environmental factors, such as temperature, humidity, and solar radiation. This modeling approach is based on the laws of thermodynamics, a set of fundamental principles that describe the physical behavior of systems involving heat and energy[20]. Before presenting the proposed model for indoor temperature forecasting, this work introduces some essential concepts of thermodynamics utilized in this study.

U-Value

The U-value, also known as the heat transfer coefficient, is a measure of heat transfer through a material or system of materials. It is defined as the quantity of heat lost or gained through a given surface area per unit of the temperature difference between the two sides of the material or system. The U-value is measured in units of Watts per square meter per Kelvin, W/m^2K, and is directly proportional to the thermal conductivity of the material[21, 22].

The U-value is inversely proportional to the thermal resistance, or R-value, which measures a material or system's ability to resist heat transfer[21]. Hence, materials with a lower U-value are better at insulating and preventing heat loss or gain. As a result, U-values are often used in building design to ensure energy efficiency. For example, the U-value of a single-glazed window is typically about 5.8 W/m^2K, while the U-value of a triple-glazed window ranges from around 1.7 W/m^2K down to around 0.4 W/m^2K[23]. This demonstrates that triple-glazed windows are far more effective at insulating than single-glazed windows, making U-values a critical factor in indoor temperature forecasting.

Heat Capacity

Heat capacity is another fundamental concept in building temperature modeling. The heat capacity quantifies the ability of a material to store and exchange thermal energy. It is defined as the quantity of heat required to raise the temperature of a substance by a certain amount, typically in the unit of J/K[24].

In building temperature modeling, the specific heat capacity is often more relevant. The specific heat capacity is defined as the heat capacity per mass, with the SI-unit J/kgK. In other words, the specific heat capacity quantifies the ability of a material to store and exchange thermal energy per its mass[24]. Therefore, this coefficient plays a crucial role in modeling and accurately forecasting the indoor temperatures of a building.

2.2.1. Building Temperature Forecasting Model

Several methods can be used to calculate transient heat transfer in buildings. The most used methods can be classified as follows[25].

1. Explicit solution of the heat diffusion equation, by finite difference and control volumes, or response function methods.
2. Model reduction techniques.
3. Model simplification techniques, such as a resistance-capacitance(RC) network.

The explicit solution by finite differences and control volumes is described in detail in the book "Energy Simulation in Building Design" by Joseph Clarke[26], upon which the building energy performance simulation software of ESP-r is based. This method involves dividing the continuous system into discrete nodes at preselected points of interest, then developing conservation equations for each node in terms of its surrounding nodes. The equations are then solved simultaneously for successive timesteps to obtain the future states of the nodal variables[26].

Clarke also suggests using an electrical RC network to better visualize and understand the system[26]. In the RC network approach of modeling building systems, the building is represented as an electrical network of time-dependent resistances and capacitances subject to time-dependent potential differences. Many authors commonly use this approach because it allows for easy visualization of the thermal simulation process[25].

In this model, nodes represent different construction elements, rooms, or glazing systems and are characterized by their capacitance. The connections between nodes represent thermal connections and are characterized by their conductance. The resulting currents in each network branch are equivalent to the heat flows between different building parts. Nodes denote state variables, in this case, temperature, which is analogous to voltage in an electrical circuit[25].

Since nodes have different capacitances, the problem is inherently dynamic, with each node responding at a different rate as it competes with its neighbors to capture, store, and release energy analogous to current in an electrical circuit. This distributed dynamic behavior, accompanied by the non-trivial nature of the branch flow and network parameters, makes building temperature modeling a complex task. The number of nodes in

the model depends on the analysis objectives and can vary widely. Generally, the more nodes included in the model, the more detailed and accurate the simulation will be[25]. However, increasing the number of nodes also increases the complexity of the model, making it more challenging to solve. Therefore, it is crucial to strike a balance between accuracy and complexity when constructing an energy model for building simulation[26]. Using this electrical network approach, the system can be visualized as in fig. 2.1, which makes it easier to understand the interactions between different parts of the system.

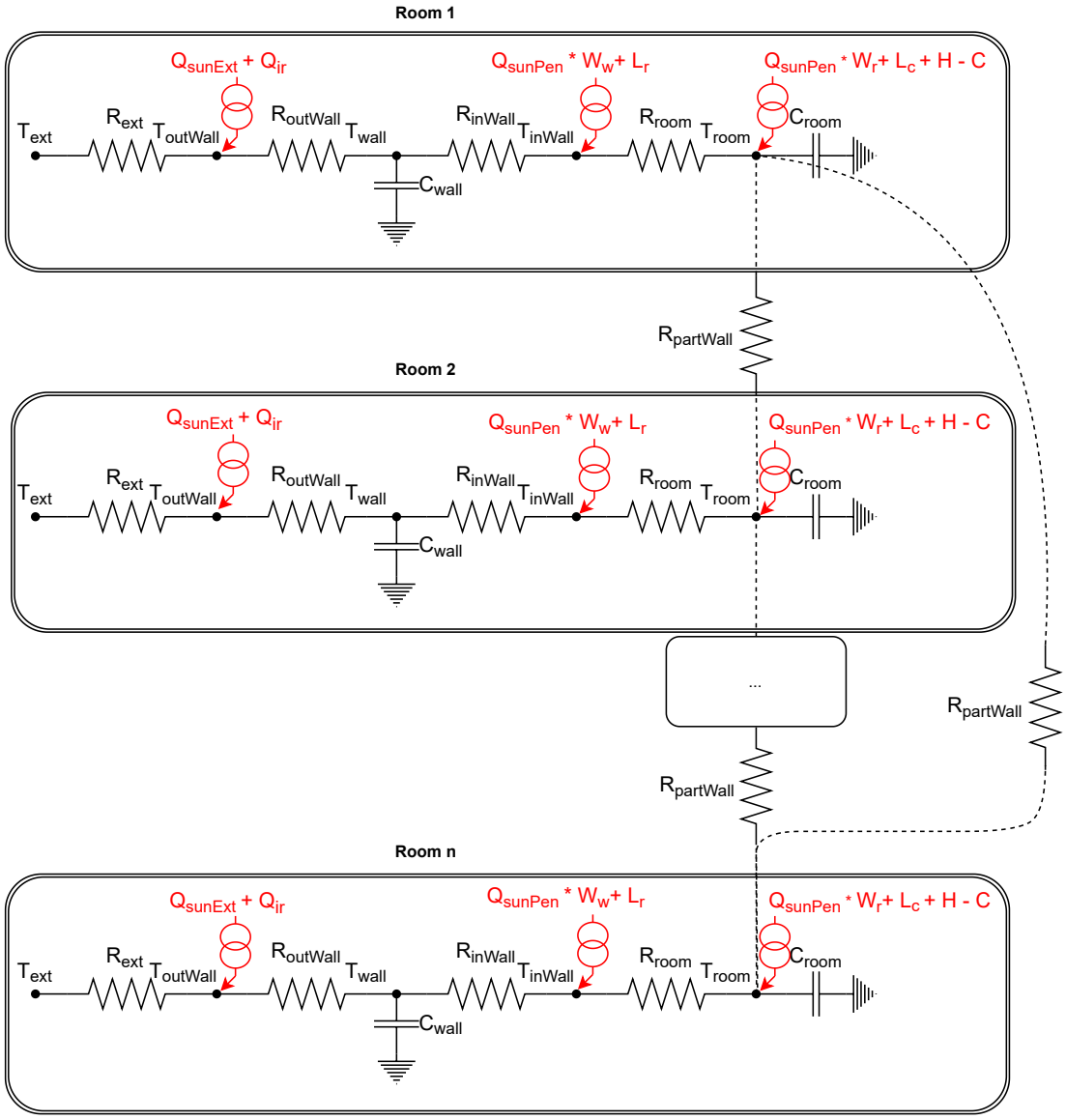


Figure 2.1.: Thermal PBM depicted as an electric circuit

Abbreviation	Explanation
T_{ext}	Temperature of the outdoor air[°C]
R_{ext}	Conductance of the thin air layer at the exterior wall exterior surface[W/K]

Continued on next page

Abbreviation	Explanation
$T_{outWall}$	Temperature of the outer part of the exterior wall[°C]
$R_{outWall}$	Conductance of the outer part of the exterior wall[W/K]
T_{wall}	Temperature of the middle part of the exterior wall[°C]
C_{wall}	Heat capacity of the heavy wall material of the room[J/K]
T_{inWall}	Temperature of the inner part of the exterior wall[°C]
R_{inWall}	Conductance of the inner part of the exterior wall[W/K]
T_{room}	Temperature of the air in the room[°C]
R_{room}	Conductance of the thin air layer at the exterior wall interior surface[W/K]
C_{room}	Heat capacity of the air and furniture in the room[J/K]
$R_{partWall}$	Conductance of the wall separating the rooms[W/K]
$C_{partWall}$	Heat capacity of the wall separating the rooms[J/K]
Q_{sunExt}	Energy flux from the sun on the exterior wall exterior surface[W]
Q_{sunPen}	Energy flux from the sun that penetrates into the room[W]
Q_{ir}	Energy flux due to longwave radiation exchange between the exterior wall surface and the surroundings[W]
W_w	Fraction of Q_{sunPen} that goes on the T_{inWall} node[W]
W_r	Fraction of Q_{sunPen} that goes on the T_{room} node[W]
L_r	Heating radiative load[W]
L_c	Heating convective load[W]
H	Heating load[W]
C	Cooling load[W]

Table 2.2.: Explanation of abbreviations in fig. 2.1

A mathematical model for an n-node system can be described with the following governing differential equation.

$$C \cdot \dot{\vec{T}}(t) = A(t) \cdot \vec{T}(t) + \vec{u}(t) \tag{2.1}$$

where $\vec{T}(t)$ denotes the temperature vector at each node, $\dot{\vec{T}}(t)$ denotes its time derivative, $\vec{u}(t)$ denotes the source term for each node, C denotes the positive diagonal thermal capacity matrix and $A(t)$ denotes the symmetric heat transfer matrix[25].

This system can be solved using various methods, including explicit and implicit Euler methods, modal spectral methods, and Fourier series methods. It can also be depicted as the electric circuit shown in fig. 2.1, with the corresponding explanation of abbreviations in table 2.2[25].

In this model, the temperature of the outer part of the exterior wall, $T_{outWall}$, takes into account the short- and long-wave radiant exchanges at the exterior surface. The term R_{ext} includes both conductive and convective terms, where the convective term is influenced by wind speed, wind direction, and outdoor temperature. Therefore, R_{ext} is time-dependent. The layers of the exterior walls have a total capacitance of C_{wall}[25]. Although these values may vary from room to room, the structure of the model remains the same for all rooms.

The inter-room connections denoted $R_{partWall}$, short for partition wall, model thermal connections between the rooms. As shown in fig. 2.1, one room may have thermal

connections with several rooms. However, this method does not account for the heat capacity of the partition wall. If the heat capacity of the partition is significant, this issue can be addressed by subdividing the heat capacitance of the wall and allocating the subdivided parts to the capacitances C_{wall} and C_{room} of the respective rooms, as done in eq. (2.2). It is important to notice that fig. 2.1 does not consider the conductance to the external environment through open windows, leakage, and ventilation, i.e., the facade is assumed to be airtight. This contrasts the model proposed by Kämpf and Robinson[25].

If the heat capacitance of the partition wall is denoted as $C_{partWall}$ and Kirchhoff's current law is used at each node, eq. (2.1) can now be rewritten as follows. Q_{ir} depends on $T_{outWall}$ to the power of four but can be linearized using a first-order Taylor expansion[25].

$$\begin{pmatrix} C_1 & 0 \\ 0 & C_2 \end{pmatrix} \cdot \begin{pmatrix} \dot{\vec{T}}_{\text{room}}(t) \\ \dot{\vec{T}}_{\text{wall}}(t) \end{pmatrix} = \begin{pmatrix} D & E \\ F & G \end{pmatrix} \cdot \begin{pmatrix} \vec{T}_{\text{room}}(t) \\ \vec{T}_{\text{wall}}(t) \end{pmatrix} + \begin{pmatrix} \vec{u}_{\text{room}}(t) \\ \vec{u}_{\text{wall}}(t) \end{pmatrix} \tag{2.2}$$

with the following properties.

$$(C_1)_{ij} = \begin{cases} C_{room,\text{i}} + \frac{1}{4}\sum_j C_{partWall,\text{ij}} & \text{if } i = j \\ 0 & \text{elsewhere} \end{cases},$$

$$(C_2)_{ij} = \begin{cases} C_{wall,\text{i}} + \frac{1}{4}\sum_j C_{partWall,\text{ij}} & \text{if } i = j \\ 0 & \text{elsewhere} \end{cases},$$

$$(D)_{ij}, = \begin{cases} -\kappa_{2,i} - \sum_j R_{partWall,\text{ij}} & \text{if } i = j \\ R_{partWall,\text{ij}} & \text{elsewhere} \end{cases},$$

$$(E)_{ij} = -(F)_{ij} = \begin{cases} \kappa_{2,i} & \text{if } i = j \\ 0 & \text{elsewhere} \end{cases},$$

$$(G)_{ij} = \begin{cases} -\kappa_{2,i} - \kappa_{1,i}(t) & \text{if } i = j \\ 0 & \text{elsewhere} \end{cases}.$$

$\kappa_1(t)$ and κ_2 denotes the conductance between the nodes $T_{ext}(t)$ and T_{wall}, and T_{room} and T_{wall}, respectively, and can be written as follows[25].

$$\kappa_1(t) = \left(\frac{R_{\text{ext}}(t) \cdot R_{\text{outWwall}}}{R_{\text{ext}}(t) + R_{\text{outWall}}}\right), \quad \kappa_2 = \left(\frac{R_{\text{room}} \cdot R_{\text{inWall}}}{R_{\text{room}} + R_{\text{inWall}}}\right).$$

The vector $\vec{T}_{\text{room}}(t)$ contains all the room temperatures of the n rooms, and $\vec{T}_{\text{wall}}(t)$ contains the wall temperatures at the n rooms.

The source term for the wall and room can be described as

$$\begin{pmatrix} u_{\text{room}}(t) \\ u_{\text{wall}}(t) \end{pmatrix} = \begin{pmatrix} \frac{\kappa_2}{R_{\text{inWall}}} \cdot (Q_{\text{sunPen}}(t) \cdot w_{\text{w}} + L_{\text{r}}(t)) \\ \frac{\kappa_2}{R_{\text{room}}} \cdot (Q_{\text{sunPen}}(t) \cdot w_{\text{w}} + L_{\text{r}}(t)) \end{pmatrix} + \begin{pmatrix} Q_{\text{sunPen}}(t) \cdot w_{\text{r}} + L_{\text{c}}(t) + H(t) - C(t) \\ \frac{\kappa_1(t)}{R_{\text{ext}}(t)} \cdot (R_{\text{ext}}(t) \cdot T_{\text{ext}}(t) + Q_{\text{sunExt}}(t) + Q_{\text{ir}}(t)) \end{pmatrix}.$$

In order to use this model to forecast the future state of the system, eq. (2.2) has to be solved for $\dot{\vec{T}}$. $\dot{\vec{T}}$ is the only unknown left in the system, and several methods can be used to solve it. Once $\dot{\vec{T}}$ is obtained, the temperature at the next timestep can be obtained by integrating or simply adding $\dot{\vec{T}}$ to $\vec{T}$ with a given step length, using, for example, the Euler method.

2.3. Data-Driven-Based Building Temperature Modeling

Data-driven-based building temperature modeling, or data-driven building temperature modeling, is an approach to indoor temperature forecasting that relies on data-driven methods such as machine learning to forecast a building's temperature evolution accurately. This approach has gained significant attention in recent years due to breakthroughs in the machine learning paradigm and the need for energy-efficient buildings to reduce energy consumption and carbon emissions.

Traditionally, indoor temperature forecasting has relied on PBMs that use mathematical equations to forecast a building's temperature evolution, as seen in section 2.2. However, such models require substantial input data, such as building geometry, material properties, and Heating, Ventilation, and Air Conditioning (HVAC) system specifications, which are often hard to derive. These models also have several limitations, such as being time-consuming, complex, and neglecting unknown physics. In addition, they may not capture the variability in building performance due to factors such as occupant behavior and weather conditions.

In contrast, data-driven building temperature modeling uses machine learning algorithms to analyze large datasets of a building's temperature evolution and environmental conditions to generate models that accurately forecast temperature evolution. These models can account for both known and unknown physics and can be used to optimize and provide recommendations for energy-efficient building operations.

Overall, data-driven building temperature modeling is a promising approach to indoor temperature forecasting that has the potential to revolutionize the way buildings are operated. Furthermore, with the increasing availability of large datasets and advances in machine learning algorithms, more accurate and effective models for optimizing building energy performance can be expected, which can help reduce energy consumption and carbon emissions, leading to a more sustainable future.

2.3.1. Data Preprocessing

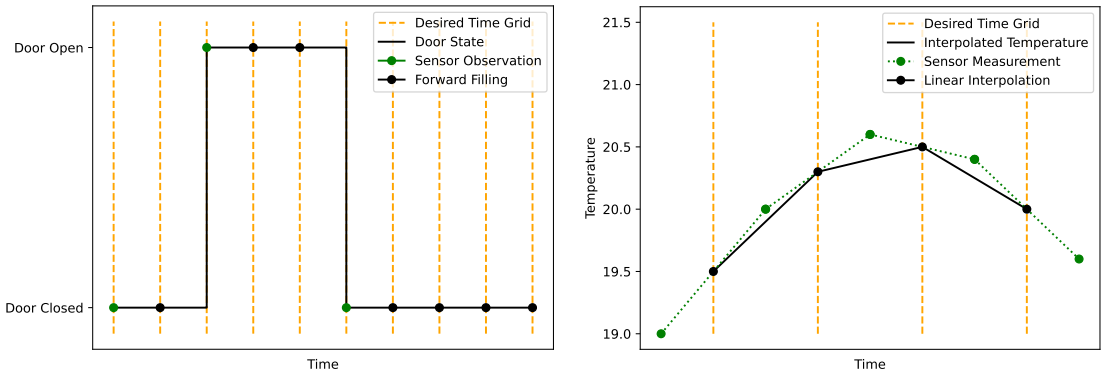
Data preprocessing is a pivotal part of the machine learning pipeline. Not only does it transform raw data to a more manageable format, but it also eases the detection of patterns and comparisons between data, thereby directly impacting the performance of the model[27]. Many preprocessing techniques exist, such as data cleaning, data reduction, data interpolation, data normalization, and data windowing. This subsection describes the key concepts used in this study. A reader familiar with these concepts is advised to skip this part, as it may be found redundant.

Data Interpolation

Data interpolation is an important statistical method in machine learning used to estimate unknown values within the range of a discrete set of known data points. Its essence lies in filling the gaps within the dataset, and resampling sensor measurements to a coordinated time grid, thereby reducing noise and enhancing the dataset's overall quality.

Forward filling is one of the simplest existing interpolation techniques. However, it describes some discrete events in a time series entirely[28]. A great example of such an event is the opening and closing of doors, monitored by a proximity sensor that samples the event of absence or presence. If a proximity sensor samples the presence of a door, the door stays closed until the proximity sensor samples its absence, and therefore the value of the closed door has to be forward filled until the door has been opened. Figure 2.2a depicts this principle. Note that the sensor observations in this figure have already been resampled to a manageable time grid before the forward filling takes place.

Linear interpolation is another simple yet prevalent interpolation technique. It simply determines the unknown value by drawing a straight line between the previous and next known value and determines the unknown value from the point of intersection with the desired time grid[28]. While this technique may not always provide the most accurate estimates due to its assumption of a constant rate of change between points, it is still feasible for slow systems with frequent sampling. Higher-order polynomial interpolation can indeed model more complex trends in the data, but it may also lead to overfitting. The principle behind linear interpolation is depicted in fig. 2.2b.



(a) Forward Filling (b) Linear Interpolation

Figure 2.2.: Principles behind the interpolation techniques

Data Normalization

Data normalization, a fundamental data preprocessing technique in machine learning, is employed to transform independent features to a similar scale. Normalization is crucial when training a DDM to achieve higher accuracy and reduce the time needed for training[29]. There are several different normalization techniques, and this work will present two of the most commonly used techniques, namely min-max normalization and z-score normalization.

Min-max normalization is a linear scaling that scales the data according to predefined upper and lower bounds, usually 0 to 1 or -1 to 1. The following equation describes the transformation process.

$$x'_{i,n} = \frac{x_{i,n} - \min(x_i)}{\max(x_i) - \min(x_i)}(n\,\text{Max} - n\,\text{Min}) + n\,\text{Min} \tag{2.3}$$

where $x'_{i,n}$ and $x_{i,n}$ respectively denote the normalized and original feature value, $\min(x_i)$ and $\max(x_i)$ denote the minimum and maximum value of the i^{th} feature, respectively, and $n\,\text{Max}$ and $n\,\text{Min}$ denote the desired upper and lower bounds of the transform, respectively[30].

Despite its benefits, min-max normalization may not be appropriate when the data contains outliers, as these can distort the rescaling. In such scenarios, more suitable options may include robust scaling methods or other preprocessing techniques[31].

Z-score normalization is another linear scaling that scales the data according to its mean and standard deviation to achieve zero mean and unit variance. Mathematically, this can be expressed as

$$x'_{i,n} = \frac{x_{i,n} - \mu_i}{\sigma_i} \tag{2.4}$$

where $x'_{i,n}$ and $x_{i,n}$ respectively denote the normalized and original feature value, and μ_i and σ_i denote the mean and standard deviation of the i^{th} feature, respectively[30].

Z-score normalization handles outliers more effectively than most other scaling methods[30]. However, it does assume that the data is normally distributed, which may not always be the case[32].

Data Windowing

Data windowing, or sliding window, is often a necessary preprocessing step in machine learning for time-series analysis. It involves partitioning the dataset into subsets or "windows" of consecutive data points. These windows, which slide over the data in a predetermined increment, help preserve the data's sequential order and allow the learning algorithm to capture temporal patterns within the defined interval[33]. This principle is depicted in fig. 2.3.

The window size, which refers to the length of consecutive data points in a window, is an important parameter. A too-small window size may fail to capture relevant long-term dependencies, while a too-large window size may dilute short-term dependencies[33]. Therefore, selecting an appropriate window size, hence appropriate feature and label sizes, is essential for the model's efficiency.

Windowing is particularly important in deep learning models such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformers, where it is used to define input sequences and output labels for the forecasts[34].

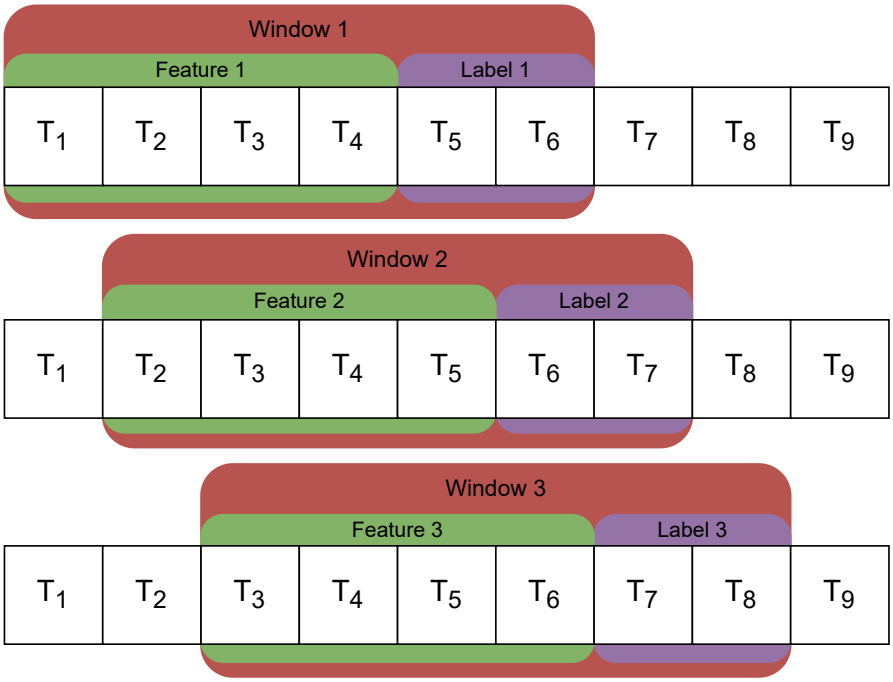


Figure 2.3.: Principle behind data windowing

Dimensionality Reduction

Dimensionality reduction is an essential step in time series analysis. It aims to reduce the complexity of the data, thereby preventing overfitting, providing simplicity, and enhancing computational efficiency. This process transforms high-dimensional data into a lower-dimensional space while preserving essential characteristics[35]. One of the goals of dimensionality reduction is to avoid the infamous "curse of dimensionality"[36].

Principal Component Analysis (PCA), a commonly used linear dimensionality reduction technique, transforms the data into a new set of orthogonal features known as principal components. These components are directions in feature space along which the original data varies the most, with the first principal component accounting for the largest possible variance in the data, the second principal component accounting for the second largest variance, and so on [35].

Another more straightforward dimensionality reduction technique is to select features based on correlation analysis. A correlation analysis is a statistical method used to assess the strength and direction of the linear relationships between pairs of features. A correlation matrix is a square table that contains the correlation coefficients for different features. The values in the matrix range between -1 and 1, where 1 represents a perfect positive correlation, -1 represents a perfect negative correlation, and 0 implies no linear correlation[37].

Highly correlated features do not complement each other, and using both as input features only increases complexity and computational cost. Removing such redundant features might increase the efficiency and accuracy of the model[37]. However, it is essential to note that the correlation matrix only captures linear relationships between variables, and non-linear relationships and dependencies require other statistical techniques.

Spectral Analysis

Spectral analysis is a powerful technique for analyzing time series data, providing insights into the periodic components and underlying frequencies that form the data. This method can reveal patterns not readily apparent in the time domain, thereby offering a deeper understanding of the data's structure[38].

Spectral analysis utilizes Fourier analysis, decomposing the time series data into sine and cosine functions of distinct frequencies. This results in a spectrum or periodogram, a plot of the power or significance of the different frequencies that highlights the dominant cycles in the data. Spectral analysis is often used for feature engineering and is particularly interesting when dealing with data that exhibits cyclical or seasonal behavior, such as temperature data[38].

2.3.2. Machine Learning Fundamentals

The field of machine learning fundamentals is vast and complex. This section will, therefore, only present the most important concepts used in this work. A reader familiar with these concepts is advised to skip this part, as it may be found redundant.

Feed-Forward Neural Network - FFNN

Feed-Forward Neural Network (FFNN) is a fundamental neural network architecture where information travels in a forward manner from the input layer, through hidden layers, to the output layer, with no recurrence, feedback, or loops[39]. Given its widespread use and straightforward design, this thesis refrains from an in-depth exploration of this topic. Nevertheless, readers seeking comprehensive understanding are encouraged to refer to "Deep Learning" by Goodfellow et al.[39].

L1 and L2 regularization

L1 and L2 regularization are widely employed and prevalent techniques to prevent overfitting in machine learning models by adding a penalty term to the loss function[39].

L1 regularization, also known as Lasso regression, introduces a penalty term that is proportional to the absolute value of the parameters. This promotes sparsity in the learned model as it tends to make some of the parameters exactly zero, effectively eliminating insignificant features and contributing to feature selection[39].

L2 regularization, or Ridge regression, adds a penalty term proportional to the square of the parameters. It encourages the parameters to be small but does not make them zero, resulting in models that are less sensitive to individual features[39].

Dropout

Dropout is another regularization technique for neural networks that operates by randomly setting a fraction of the output from the neurons in each layer to zero during training, which helps to prevent overfitting. Dropout prevents neurons from co-adapting too much during training, which negatively impacts the model's generalization.

The underlying intuition is that this approach forces the network to learn more robust features useful in conjunction with various random subsets of the other neurons. Despite its simplicity, dropout has been shown to produce significant improvements in the performance of neural networks on supervised learning tasks[40].

Early Stopping

Early stopping is a third regularization technique used to prevent overfitting in machine learning models during the training of neural networks. As model training advances, the training error starts to decrease. The validation error, the error on unseen data, also initially decreases, but at some point, it will reach a minimum and start to increase, indicating overfitting. Early stopping intervenes at the point when validation error begins to rise, ending the training process to ensure that the model generalizes well to unseen data. This technique eliminates the need for setting and tuning an arbitrary number of training epochs and reduces the risk of overfitting without adding computational complexity or requiring additional hyperparameter tuning[41]. The principle behind early stopping is depicted in fig. 2.4.

While early stopping has proven effective, it is important to know its limitations. For instance, the optimal stopping point can vary depending on the dataset and model architecture, and early stopping may lead to underfitting if not appropriately tuned. Several ways exist to decide on the optimal stopping point, with the simplest being to set a parameter called "patience". This "patience" parameter decides the acceptable number of epochs without improving the best validation loss before the training is terminated. The model can then be reverted to the model which performed the best validation loss[41].

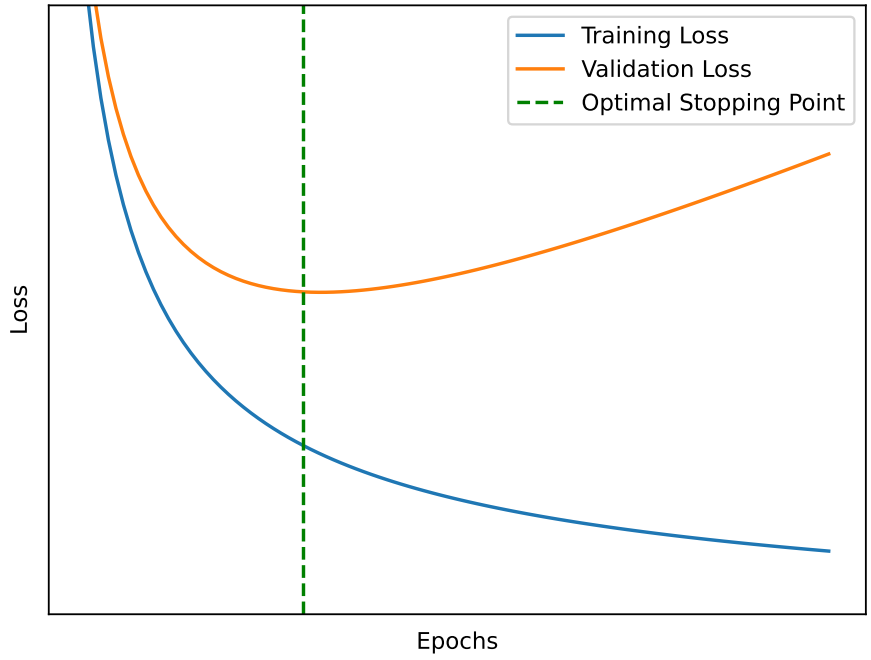


Figure 2.4.: Principle behind early stopping

Rolling Forecast

Essentially, rolling forecast is a forecasting type that makes it possible for the model to forecast longer into the future by using its own previous forecasts as input into the model iteratively[42]. This method introduces some uncertainty to the input of the predictive model, potentially increasing the risk of divergence. However, it has proven effective in many cases and is often used in time-series forecasting. Pseudocode for rolling forecasting is given in algorithm 1.

Algorithm 1 Rolling Forecast

```
Input: prev_n_timesteps, n_iterations, forecast_horizon_length
for i in range(n_iterations) do
    forecast = PredictiveModel(prev_n_timesteps, forecast_horizon_length)
    Append the n=forecast_horizon_length timesteps of forecast to prev_n_timesteps
    Remove the n=forecast_horizon_length first elements from prev_n_timesteps
end for
```

2.3.3. Long Short-Term Memory - LSTM

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) architecture that Hochreiter and Schmidhuber introduced in 1997[43]. The LSTM architecture is designed to overcome the vanishing gradient problem in traditional RNNs. The vanishing gradient problem occurs when the gradient signal in the backpropagation algorithm exponentially decreases as it propagates back through time, causing the weights to be updated very slowly or not at all[44, 45].

LSTMs aim to solve the vanishing gradient problem by utilizing the cell state and forget gate concept. The cell state is depicted as the horizontal line running through the top of fig. 2.5, denoted C, acting as the conveyor belt of the LSTM. It runs through the entire chain and is only altered by minor linear interaction[45]. This makes the information able to flow more or less unchanged down the line.

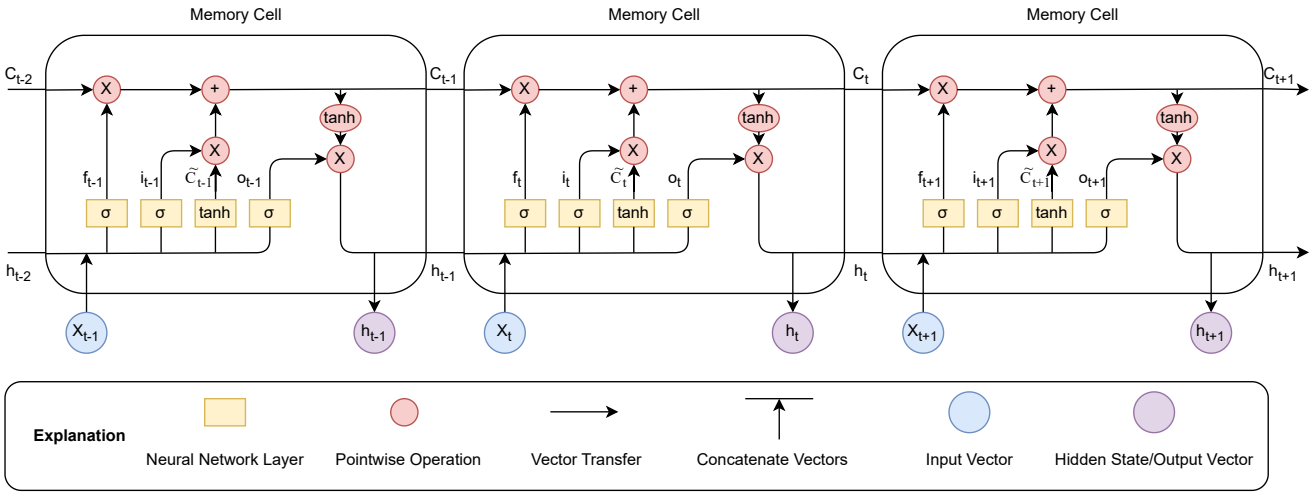


Figure 2.5.: The LSTM architecture with three memory cells

The only structures that can alter the cell state are the gates, namely the forget gate, the input gate, and the output gate. The forget gate, denoted f in fig. 2.5, consists

of the output from the previous timestep, often referred to as the hidden state, h_{t-1}, concatenated with the current state, x_t. This vector is then processed by the trained sigmoid layer, σ, which outputs a number between zero and one for each state in the cell state before these values are pointwise multiplied with the previous cell state[45]. If the sigmoid layer outputs zero for a state, it implies forgetting everything about this state, while one implies remembering everything. Mathematically, this can be expressed as

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right). \tag{2.5}$$

where W_f and b_f denote the weight and biases of the corresponding sigmoid layer.

The input gate, denoted i in fig. 2.5, decides what new information will be stored in the cell state. A trained sigmoid layer processes the concatenated states, $[h_{t-1}, x_t]$, before the values are pointwise multiplied with the same states processed by a tanh layer. The tanh layer creates a vector of candidate values, $\tilde{C}_t$, that could be added to the cell state, and the sigmoid layer decides how much of each value will be added to the cell state[45]. Mathematically, this reads as

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \end{aligned} \tag{2.6}$$

where W_i and b_i denote the weight and biases of the corresponding sigmoid layer, and W_C and b_C denote the weight and biases of the corresponding tanh layer.

The output from the pointwise multiplication of i and $\tilde{C}$ is then added to the output from the forget gate to form the new cell state. The updated cell state can be expressed as

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t. \tag{2.7}$$

The output gate does not directly alter the cell state but is responsible for deciding what the memory cell will output and feed into the next memory cell in the chain, thus influencing the cell state. The output is essentially a filtered version of the new cell state. The cell state is first put through a tanh operation to squeeze the values between -1 and 1. A trained sigmoid layer processes the concatenated states, $[h_{t-1}, x_t]$, and pointwise multiplies them with the output from the tanh operation to decide which parts of the cell state will be outputted[45]. This can be expressed mathematically as

$$\begin{aligned} o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned} \tag{2.8}$$

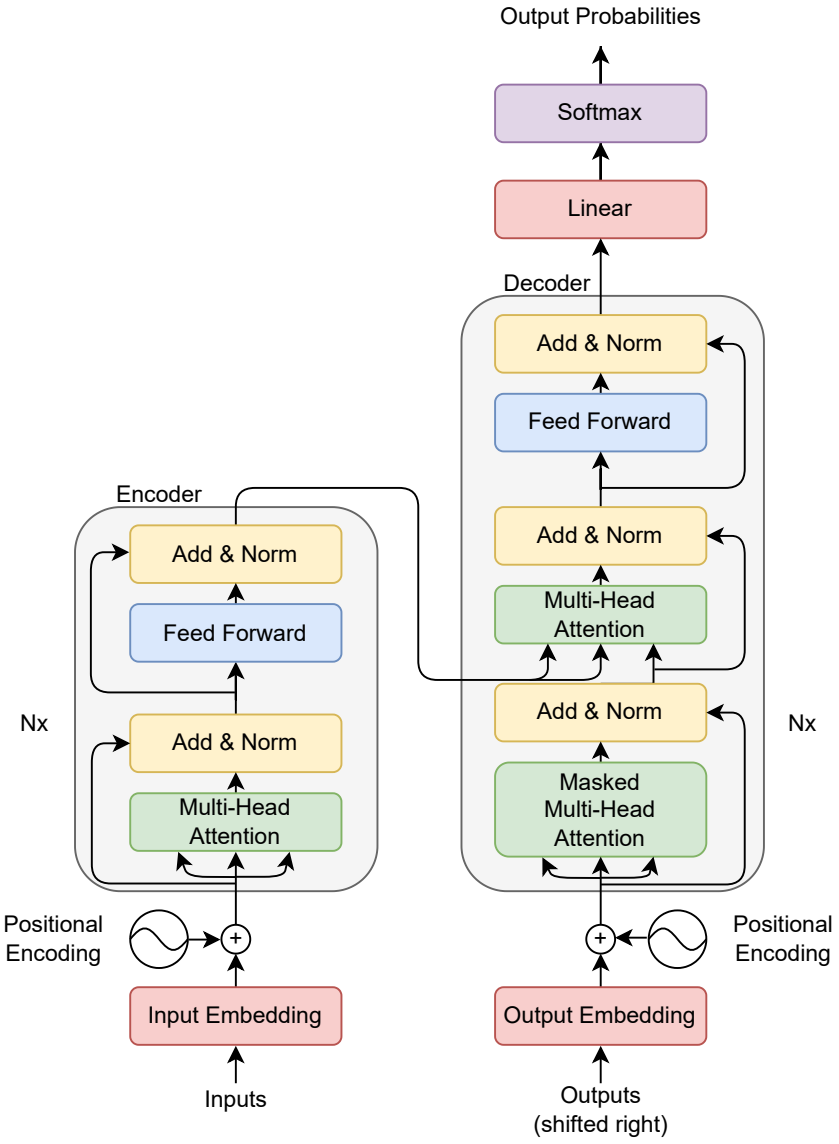
where W_o and b_o denote the weight and biases of the corresponding sigmoid layer.

In the context of building temperature modeling, LSTMs can be used to forecast a building's temperature evolution based on historical temperature data and environmental conditions. By training an LSTM model on historical temperature data from the asset and its surroundings, the LSTM should be able to learn the dynamics of the temperatures of the asset and make reasonable forecasts of the temperature evolution. In addition, the recurrent properties of the LSTM should be capable, to a certain extent, of capturing various factors such as the presence of people, heat from a fireplace or radiators, and other elements that influence the temperature evolution of the asset.

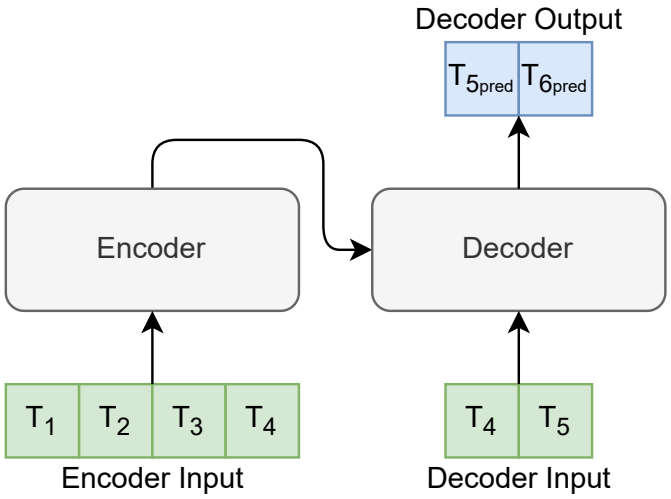
2.3.4. Transformer

The Transformer is a neural network architecture introduced in 2017 by Vaswani et al. in the paper "Attention Is All You Need"[46]. The architecture was initially proposed for Natural Language Processing(NLP) tasks such as machine translation and language modeling. It was quickly recognized as the state-of-the-art architecture for a wide range of NLP problems[47]. The Transformer architecture is the engine behind several recent breakthroughs in the AI paradigm, such as the cutting-edge, staggering chatbot released by OpenAI in November 2022 named ChatGPT[48]. However, there are many similarities between NLP and time-series forecasting, and recent studies have shown that the Transformer also can be used to model physical systems[14].

The Transformer consists of an encoder-decoder framework where the encoder and decoder comprise a stack of N identical layers. These N layers consist of multiple self-attention and feed-forward neural networks, as seen in fig. 2.6a.



(a) Depiction of the Transformer architecture, as proposed by Vaswani et al.[46]



(b) Depiction of the input and output of the Transformer during training

Figure 2.6.: The Transformer architecture

Self-attention is a mechanism that allows the network to weigh the importance of different parts of the input sequence while computing its representation. This mechanism computes a weighted sum of the input sequence, with the weights derived from a function that depends on the input sequence itself. The function is learned during training, allowing the network to focus on different parts of the sequence. The self-attention mechanism allows the model to capture long-term dependencies in a sophisticated manner[46].

The decoder additionally consists of a masked self-attention sub-layer. This masking is necessary during training and inference but is especially important during training. When the Transformer is trained, the decoder is provided with the correct outputs shifted right, as depicted in fig. 2.6b. The output is then generated autoregressively, which means the model generates the subsequent output at each step based on the previous correct outputs. In order to prevent the model from cheating by peeking ahead and generating outputs that depend on correct values from the future, the decoder input needs to be masked. This is accomplished by masking all values in the decoder input sequence after the position of the current value, which ensures that the decoder can only attend to the positions it has already generated[46].

Due to the autoregressive generation of outputs, inference with a Transformer must differ from the training. During inference, the correct outputs are inherently not known. Therefore, the autoregressive output generation must be generated based on the previously forecasted outputs instead of the correct outputs. This can be done iteratively by re-feeding the generated forecasts for each forecasted output position into the decoder until the desired forecast length is reached. For example, to forecast the first output, T_{5pred} in the context of fig. 2.6b, the encoder is provided T_1 to T_4 as done during training. However, the correct T_5 is unknown, so the decoder will only be provided with the correct T_4 for the first iteration. The decoder will then output a forecast for T_5, namely T_{5pred}, which will be added to the decoder input for the next iteration. The decoder input will now consist of T_4 and T_{5pred}, resulting in new forecasts for T_5 and T_6. The decoder input will then be updated with the new forecasts, and this procedure can be

repeated until the desired forecast horizon is reached[49].

Another essential part of the Transformer is its positional encoding. In contrast to the LSTM, which models time dependencies through its recurrent structure, the Transformer contains no recurrence. Instead, the Transformer handles time dependencies through self-attention and positional encoding. The positional encoding uniquely encodes every sample in the sequence, allowing the Transformer to evaluate the sample's impact on the output. The original paper proposed the following positional encoding based on the relative position of the sample in the sequence.

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(pos/10000^{2i/d_{\text{model}}}\right) \\ PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d_{\text{model}}}\right) \end{aligned} \tag{2.9}$$

where pos is the relative position in the sequence, i is the dimension of the positional encoding and d_{model} is the dimension of the model itself[46]. For time series forecasting, this encoding can be rewritten as a temporal encoding, where each sample's position is represented by its timestamp converted to sines and cosines with appropriate periods[50]. The positional encoding can then be rewritten to

$$\begin{aligned} PE_{(t,2i)} &= \sin\left(\omega_i t\right) \\ PE_{(t,2i+1)} &= \cos\left(\omega_i t\right) \end{aligned} \tag{2.10}$$

where ω_i represents the handcrafted frequencies typically set to represent daily and yearly periods.

The positional encoding allows each sample in the input sequence to be processed separately, unlike the LSTM, where the sequence must be processed sequentially since each timestep depends on the previous timestep. This allows the Transformer to perform many computations in parallel, which results in faster training, especially if GPUs are available for computation[46]. However, during inference, the model has to predict outputs autoregressively based on previous predictions, which limits its ability to perform computations in parallel. Efficient generative inference for Transformer models is currently an active field of research[49].

To prevent overfitting, the Transformer architecture applies dropout to the output of each sub-layer[46]. This dropout method randomly disables neurons during training to promote generalization and prevent overfitting[40].

In summary, the Transformer, with its ability to handle long-term dependencies and capture contextual information, is well-suited for modeling the intricate interactions between different components of a building's thermal system. In addition, by incorporating external factors such as weather conditions and occupant behavior, the model can accurately forecast a building's temperature evolution.

2.4. Hybrid Analysis and Modeling-Based Building Temperature Modeling

The HAM framework aims to exploit the complementary strengths of DDMs and PBMs to enhance the accuracy, reliability, and interpretability of indoor temperature forecasts. By incorporating both models, HAMs can account for the inherent complexity of building temperatures while maintaining the flexibility to adapt to changing conditions through the DDM. Several HAM strategies have been proposed in recent research, including Reduced Order Modeling, Physics-Guided Machine Learning, Data-Driven Equation Discovery, and the Corrective Source Term Approach[51].

2.4.1. Corrective Source Term Approach - CoSTA

The Corrective Source Term Approach (CoSTA) is a novel HAM approach proposed by Blakseth et al. in 2021[12]. A significant difference between CoSTA and the other previously mentioned HAM approaches is that CoSTA utilizes the PBM to the greatest extent possible by using it as the cornerstone of the forecast. The PBM consists of the governing equations of the system, which, as discussed in section 2.1.1, yields high interpretability and trustworthiness. However, these equations cannot provide outstanding accuracy due to several factors, such as unknown and hidden physics and assumptions made while modeling.

CoSTA seeks to bridge this inaccuracy by employing a DDM to predict the corrective source term of the governing equations of the PBM and then augment the governing equations with this corrective source term before the final output is produced. Hence, this approach retains the utmost of the interpretability of the PBM[12].

CoSTA has recently been found to outperform comparable DDM and PBM models in a series of numerical experiments on one-dimensional heat diffusion. The CoSTA model often reduced the predictive errors by several orders of magnitude compared to the other models while also generalizing better than pure DDMs. Due to its flexible yet solid theoretical foundation, CoSTA provides a great modular framework for further advancements within the modeling of dynamical systems. This theoretical foundation also guarantees its capacity to model any system governed by deterministic partial differential equations[12]. If able to model the source term accurately, CoSTA has the potential to diminish or even solve the accuracy issue of the PBM.

CoSTA and Building Temperature PBM

To incorporate the PBM proposed in section 2.2.1 into a CoSTA model, eq. (2.1) would have to be rewritten as follows.

$$\dot{\vec{T}}(t) = C^{-1}A(t) \cdot \vec{T}(t) + C^{-1} \cdot \vec{u}(t). \tag{2.11}$$

where the $A(t)$ denotes the symmetric heat transfer matrix and C denotes the diagonal matrix with the heat capacities corresponding to each room on the diagonal. The C matrix will always be invertible since a room cannot have zero heat capacity. With the added corrective source term, $\vec{\hat{\sigma}}(t)$, the equation reads as follows.

$$\dot{\vec{T}}(t) = C^{-1}A(t) \cdot \vec{T}(t) + C^{-1} \cdot \vec{u}(t) + C^{-1} \cdot \vec{\hat{\sigma}}(t). \tag{2.12}$$

The solution of the system can then be approximated with the Euler method as follows.

$$\begin{aligned}\vec{T}^{n+1} &= \vec{T}^{n} + h \cdot \dot{\vec{T}}(t^n) \\ &= \vec{T}^{n} + h \cdot (C^{-1}A(t^n) \cdot \vec{T}(t^n) + C^{-1} \cdot \vec{u}(t^n) + C^{-1} \cdot \vec{\delta}(t^n))\end{aligned} \tag{2.13}$$

where h denotes the chosen step length. The step length is a parameter that requires careful selection. A smaller step length may achieve a higher accuracy but comes at the cost of a more demanding computation to reach the desired next step.

Training and Inference

The training and inference routines of PBM, DDM, and CoSTA are depicted and compared in fig. 2.7. Note that a PBM does not need to be trained and is therefore not included in fig. 2.7a.

Training and inference procedures are more complex for a CoSTA than a DDM since the CoSTA has to calculate the output of three models before obtaining the next forecasted temperature, while the DDM only has to calculate the output of one model.

The increased complexity is apparent in the training comparison in fig. 2.7a. The CoSTA training approach will be described in a step-by-step manner in the following list.

1. The CoSTA first has to calculate the next temperature from the PBM model given the last observed temperature, $\vec{T}^{n}_{ref}$, and with no corrective source term.

2. This produces $\vec{\tilde{T}}^{n+1}$, which can be interpreted as the PBM's "best guess" of the next temperature.

3. This "best guess" is subsequently provided as input to the DDM responsible for predicting the corrective source term, $\vec{\tilde{\delta}}^{n}$. The DDM can also be provided with additional relevant features, for example, time of day and weather conditions, denoted by f^n in fig. 2.7a.

4. Once the corrective source term is obtained, the PBM is once again solved with the same input temperature, $\vec{T}^{n}_{ref}$, but this time together with the corrective source term produced by the DDM.

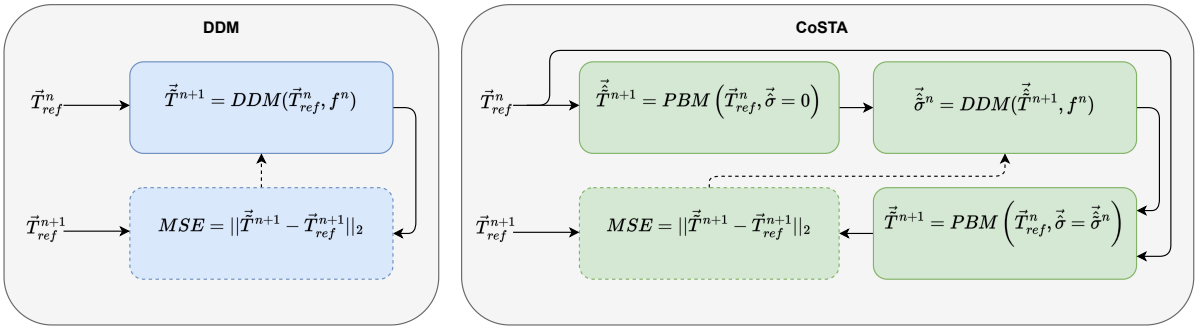
5. This output from the PBM is the final output of the CoSTA, $\vec{\tilde{T}}^{n+1}$. The final output is then compared to its label, the true temperature of the next state, in order to calculate error gradients and update the weights of the DDM.

This approach differs from the original CoSTA, as proposed by Blakseth et al.[12], in two ways. Firstly, the additional relevant features as input to the DDM, f^n, are an expansion of the original CoSTA architecture and seek to make the DDM able to make better predictions based on more relevant context.

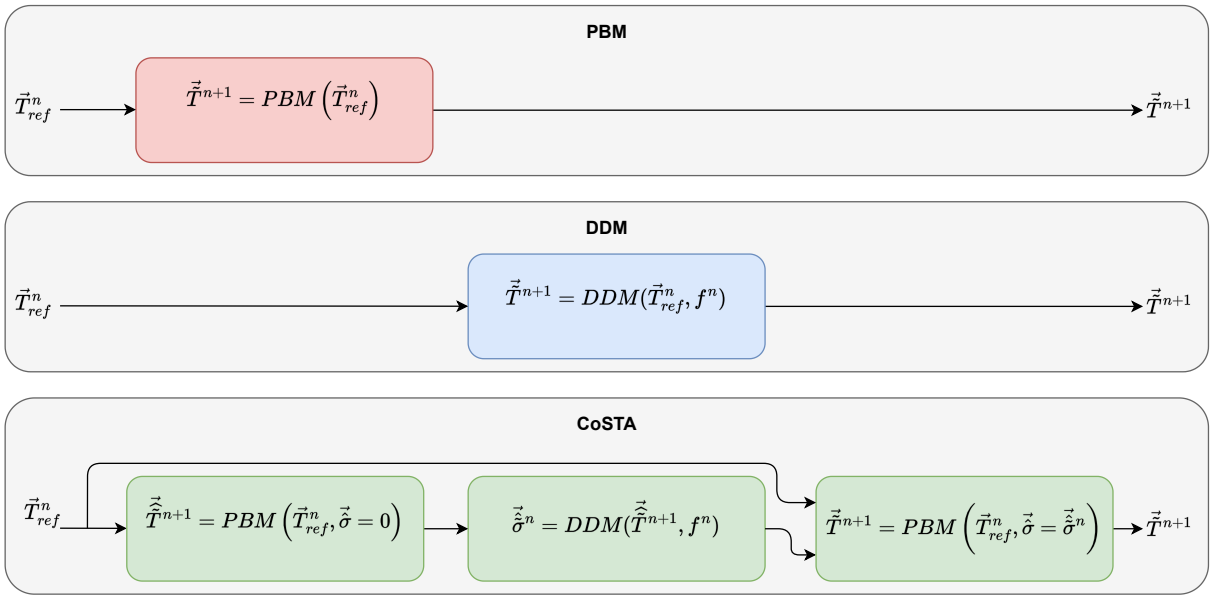
Secondly, the original approach labeled the output from the DDM, $\vec{\tilde{\delta}}^{n}$, directly with the true corrective source term in order to train the DDM. The true corrective source term can be obtained from eq. (2.13). However, by utilizing frameworks such as PyTorch[52],

which automatically creates computational graphs based on operations done on the fundamental data structures (tensors) and calculates error gradients for each parameter using the chain rule differentiation[53], this process can be altered as depicted in fig. 2.7a, which should yield the same results.

The inference routine of a CoSTA and how it differs from a PBM and DDM is depicted in fig. 2.7b. To infer a forecast from a CoSTA model, the same steps described in the training routine have to be performed. To quickly summarize, this is done by first solving the PBM without the corrective source term, then predicting the corrective source term based on this output of the PBM together with additional relevant features before the PBM is solved again, this time together with the predicted corrective source term, to produce the final output.



(a) Training procedure comparison of DDM and CoSTA



(b) Inference procedure comparison of PBM, DDM, and CoSTA

Figure 2.7.: Comparison of PBM, DDM, and CoSTA, inspired by Blakseth et al.[12]

3. Data

The data used in this study consisted of sensor measurements from 21 sensors from Disruptive Technologies, more specifically, 16 temperature sensors and five proximity sensors, placed in different locations of the asset. The temperature sensors measured the temperatures in different rooms of the asset and the outdoor temperature. The proximity sensors were responsible for monitoring the opening and closing of doors. The sensor layout is depicted in fig. 3.1.

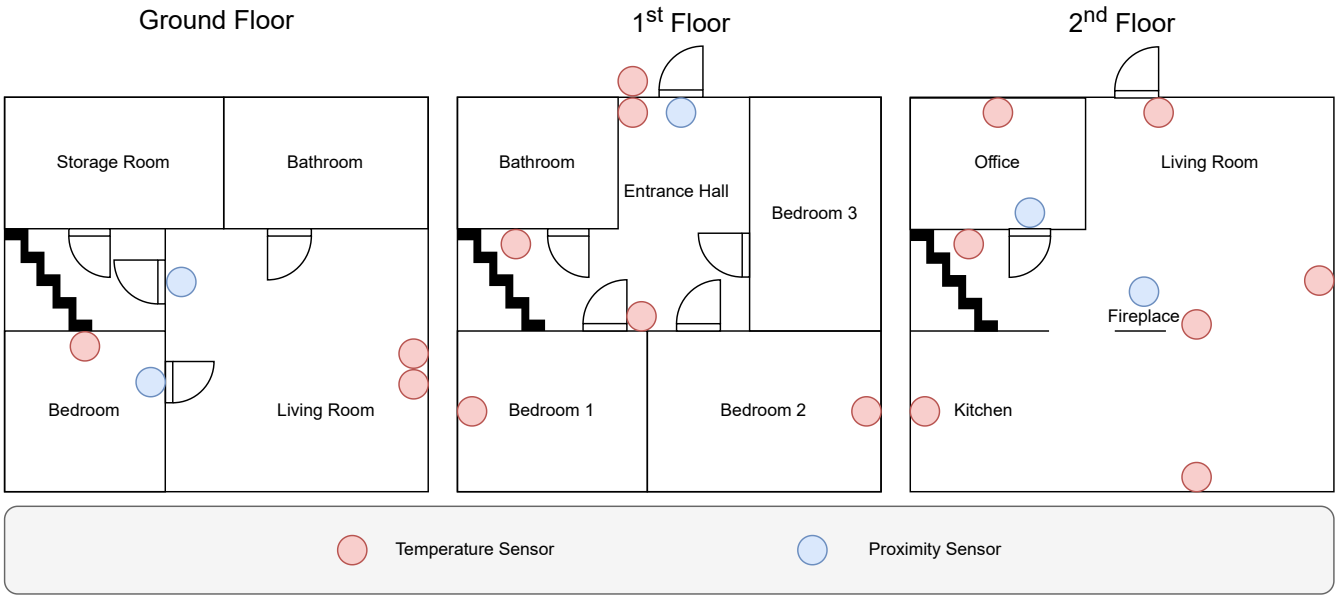


Figure 3.1.: Sensor layout at the asset

These sensors recorded data from 07.01.2022 until 19.02.2023, resulting in over a year's worth of measurements. Throughout this period, each temperature sensor measured its target with a sampling frequency of one sample every 15 minutes, leading to more than 39,000 measurements per sensor. Furthermore, the proximity sensors sampled their state every time it changed, i.e., when their door was opened or closed. Therefore, the different proximity sensors have a different number of samples based on how frequently their corresponding door was used. Nevertheless, all proximity sensors provide a complete record of the opening and closing of their corresponding doors.

3.1. Resampling and Interpolation

The first step in dealing with such time-series sensor data was to resample the data onto a manageable time grid. Even though the continuous temperature sensors all have a sampling period of 15 minutes, they did not coordinate the sampling to happen on the

same timestamp. Therefore, the samples had to be interpolated onto a common time grid. This was done by utilizing the principle of linear interpolation as demonstrated in fig. 2.2b. The thermal response of a building is relatively slow. With a sampling period of 15 minutes, linear interpolation is believed to be complex enough to capture the dynamics of the temperatures in the data.

For the discrete sensor data, i.e., the proximity sensors monitoring the doors, the sensor observations were first resampled onto the closest time grid. After that, the values for the undetermined time grids were forward-filled from the determined values, utilizing the principle depicted in fig. 2.2a. The values "Door Open" and "Door Closed" were also replaced with the values "0" and "1" to make them manageable for computing purposes.

The resampling process also revealed that one of the sensors in the entrance hall ceased recording temperatures in the middle of June 2022. The measurements from this sensor were therefore removed for further analysis.

3.2. Data Analysis

Data analysis is crucial in machine learning, especially when dealing with time series data. Through exploratory analysis, one better understands the data and its underlying structures and patterns. It identifies errors, irrelevant information, and weaknesses in the dataset, which allows for cleaning the dataset in a more sophisticated way. It also aids in feature engineering, as knowledge gained can be used to extract meaningful features from the data. This process culminates in a neat and lighter dataset, simplifying the training process and enhancing the model performance[54, 55]. The following paragraphs present the analysis conducted on this dataset.

3.2.1. Correlation Analysis

Correlation analysis is a simple yet efficient measure when analyzing time series data. An example of a correlation analysis is analyzing the correlation matrix. The correlation matrix for the temperature sensors in this dataset is depicted in fig. 3.2. A great example of redundant features is "2fBalconyEntrance" and "2fLivingRoomCenter". These sensors have a correlation of 0.99, meaning that they essentially correlate perfectly and describe the same phenomena. Using both of these features as input to the model is simply redundant and will only increase computational cost and make the optimal direction of the gradient descent less apparent during training.

For this study, "2fBalconyEntrance" and "2fLivingRoomWindow" were dropped from the dataset due to their correlation of 0.99 with "2fLivingRoomCenter", hence providing essentially the same information. "2fStair" was also dropped due to its correlation of 0.95 with "2fLivingRoomCenter", which is relatively high. Finally, "0fLivingRoomCeiling" and "0fLivingRoomFloor" have a correlation of 0.94 and were therefore squashed together to one new feature, "0fLivingRoom", providing a mean temperature for the ground floor living room. This reduction resulted in a maximum correlation of 0.92 be-

tween the features in the dataset, which is still relatively high. However, in fear of losing information, correlations less than or equal to 0.92 were kept. This process reduced the dimensionality of the dataset by 4 dimensions.

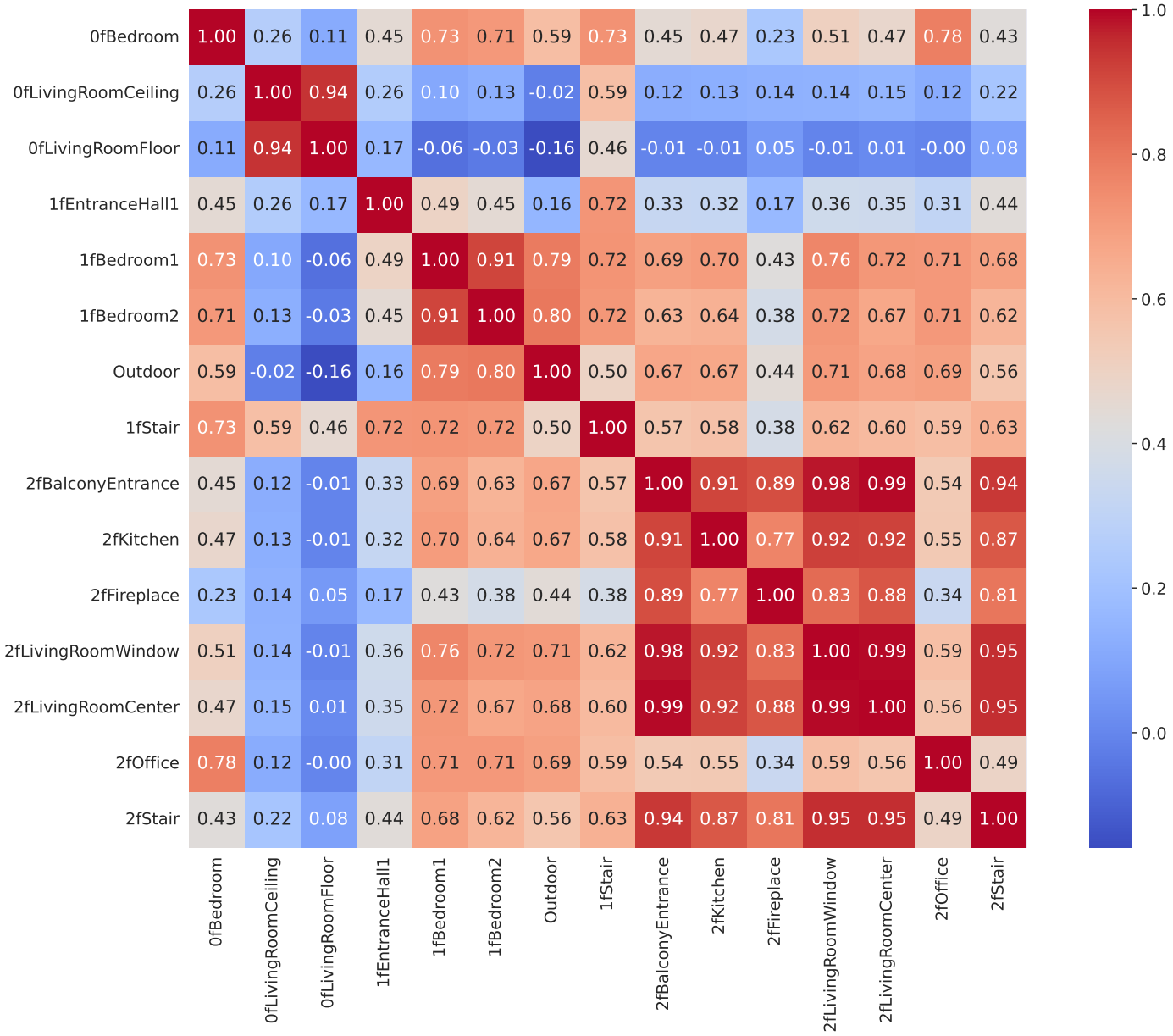


Figure 3.2.: Correlation matrix of the time series data

3.2.2. Spectral Analysis

Spectral analysis is a technique that uncovers the underlying dominant cycles in the data and is essential when analyzing time series data. When dealing with temperature data, it is often possible to make well-educated guesses of the underlying cycles in advance, as the outdoor temperature has a clear daily and yearly cycle. There might even be a weekly cycle for the inside temperature in some circumstances. However, a periodogram is a great way to assess and confirm these guesses and uncover other less apparent cycles.

A periodogram of the temperature in the second-floor living room is depicted in fig. 3.3. Here, the daily cycle is very apparent, with high power on the daily frequency. The yearly cycle also seems quite dominant. However, it is difficult to determine what goes on between the weekly and yearly cycles. A viable reason for this is that the dataset only contains measurements for just over one year, which makes it hard to determine the significance of these frequencies.

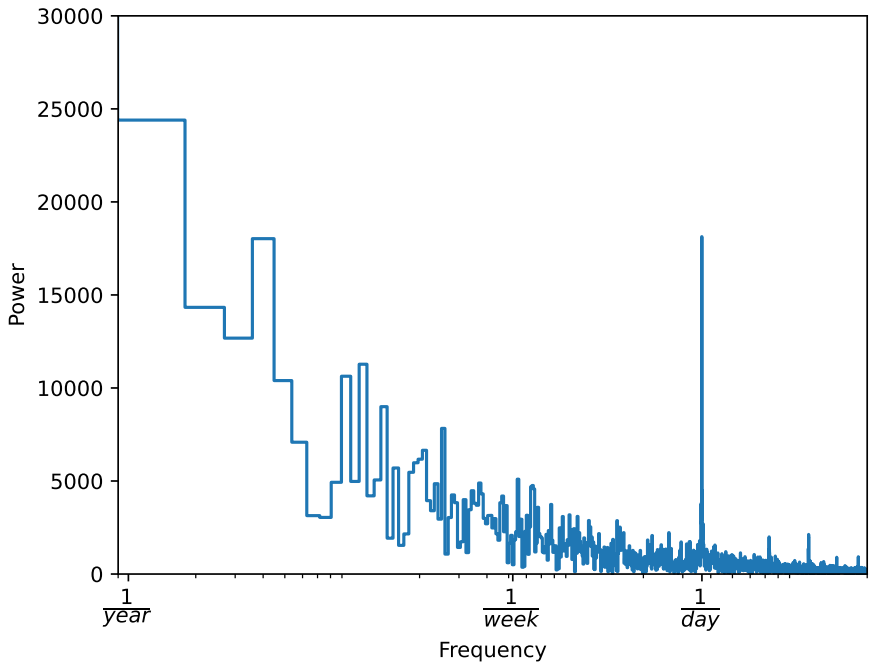


Figure 3.3.: Periodogram of the time series data

From this analysis, it was apparent that some features regarding these cycles should be engineered. The daily and yearly cycles are evident. The weekly cycle may also provide information that allows the model to distinguish between individual days of the week and between weekdays and weekends. Therefore, this cycle was also added to the features.

These features were engineered using the sine and cosine functions applied to the respective frequencies, according to eq. (3.1).

$$\begin{aligned} feature_1(t) &= \sin(\omega_i t) \\ feature_2(t) &= \cos(\omega_i t) \end{aligned} \tag{3.1}$$

where ω_i denotes the desired handcrafted frequency. For this study, the daily, weekly, and yearly frequencies were added to the dataset. These features were added to every sample in the dataset to make the model able to distinguish between the timestamps in a sophisticated manner. The yearly cycle is depicted in fig. 3.4.

Figure 3.4 also highlights the need for two trigonometric functions to model the timestamps instead of one. With only the sine, the model would not be able to distinguish between, for example, the 1st of February and the 1st of June. However, when complemented by the cosine function, which represents a sine function phase-shifted by $\pi/2$,

the model can uniquely distinguish between every timestamp throughout the year.

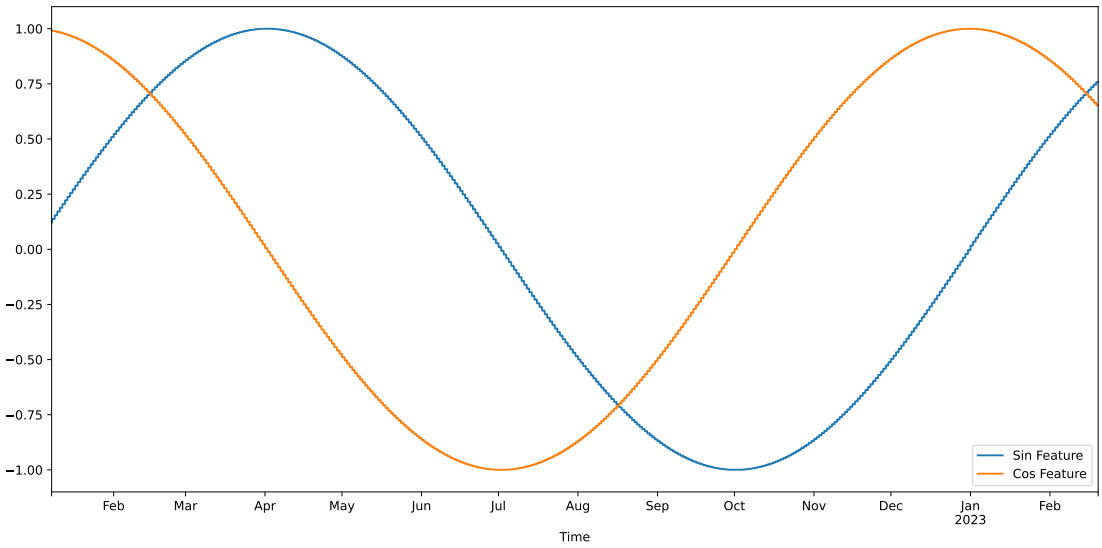


Figure 3.4.: Feature engineering of the yearly cycle

3.3. Resulting Dataset

As a result of this data preprocessing, each of the 22 remaining features in the dataset - distributed among 11 temperature sensors, five proximity sensors, and six cyclic features describing the temporal characteristics - was represented by a series of 39,166 samples. This resulting dataset is described by the box plot in fig. 3.5.

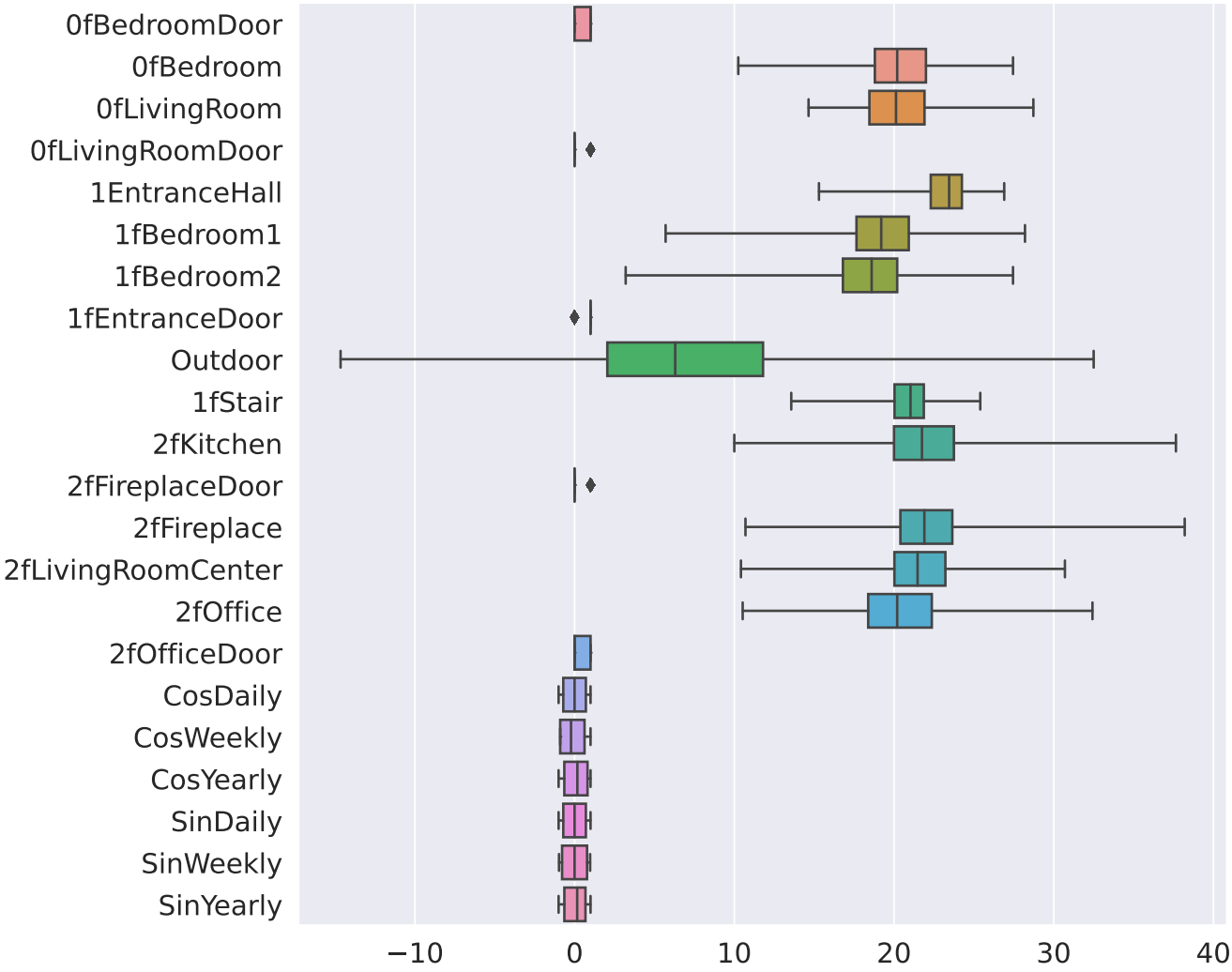


Figure 3.5.: Box plot of the resulting dataset

4. Method

The objective of the implementations introduced in this chapter was to create models from the various paradigms that accurately predicted temperature evolution in different rooms of the asset. Subsequent sections detail the methods employed for each paradigm. All models used the data presented in chapter 3, with some alterations tailored to their respective modeling paradigms, which will also be outlined.

4.1. Physics-Based Modeling - PBM

A thorough PBM of the temperatures of the specified asset was developed during the specialization project leading up to this thesis[1]. This model was implemented using ESP-r, a comprehensive, multi-domain, state-of-the-art simulation environment for building energy analysis. The method behind this model is not presented in this thesis. For further insight, the specialization report can be reviewed.

4.2. Data-Driven Modeling - DDM

During this work, two different DDMs were developed for the forecasting task. The first model utilized an LSTM architecture combined with a Feed Forward Neural Network (FFNN), while the second model employed the newer and increasingly popular Transformer architecture. Both models utilized the dataset described in chapter 3, with some modifications.

4.2.1. Data Preprocessing

Although initial data preprocessing was carried out as detailed in chapter 3, additional preprocessing was necessary for the dataset to be optimal for these machine learning architectures.

The most crucial step in the data preprocessing stage of supervised machine learning is the split of the dataset into train and test sets. This is crucial to avoid data leakage between the train and test data. The universal approximation theorem states that feed-forward neural networks with as few as one hidden layers can approximate any function[56]. Therefore, if the model is exposed to the test data during training, it might approximate it. Even if it fails to approximate it effectively, the results will still be still invalid. This is because the model validates its performance on the same data it has already been given the features and correct labels, which for obvious reasons, is

undesirable. The test data is the only way to validate the results from the model, and if there has been a data leakage between the test and train data, any results will be corrupted.

For this study, the test set was set to July 2022. This was done to get the best possible basis of comparison to the PBM developed in the specialization project[1]. This PBM neglected the influence of any internal heat source, such as radiators. Therefore, its simulation will be more accurate in summer, such as in July, when the radiators will most likely be turned off.

The choice of July as the test month may not have been optimal. The asset has sensor measurements for a year and a month, and choosing the last month of measurements as test data would lead to the model not having to extrapolate with regard to the season when assessing the model. In addition, July is known to be one of the hottest months in the specified location of the asset, meaning that the model might also have to extrapolate on the temperature when assessing its performance. Finally, DDMs are notorious for underperforming when extrapolating, and purposely extrapolating during testing will not lead to the best results accuracy-wise. Nevertheless, one may examine how the model performs during extrapolation, which may work as quality assurance.

The training set was again split into training and validation sets. The validation set was set to August 2022, while the training set kept the remainder of the data. The data split is depicted in fig. 4.1. This split resulted in the test set and validation set each comprising 8% of the total dataset, while the train set comprised the remaining 84%.

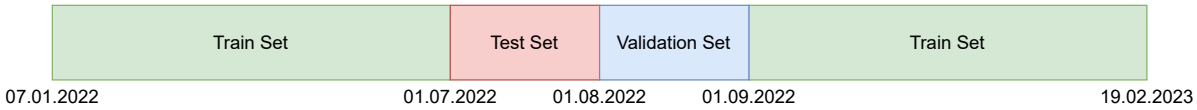


Figure 4.1.: Train, validation, and test split

Splitting validation and test sets into two separate months may not be beneficial, as it may break with the principle of independent and identical distribution (IID). However, to compare the model to the PBM, the choice of July as the test set was deemed necessary. The choice of August as the validation set was made in order to be able to tune the hyperparameters with a rolling forecast in the same manner as the model was to be evaluated on the test set. This testing procedure simulates how the model would be used in a real-life scenario and, as such, acts as an effective performance metric.

Another measure taken to make the sets more likely to fulfill the IID property, as well as make the data more stationary, was to differentiate the data. Differentiation in this context involved transforming the data to reflect temperature changes between timestamps rather than absolute temperatures, which can help minimize seasonal variation and non-stationarity. Non-stationarity in data is infamous for its impact on the accuracy and errors of forecasting in machine learning. Therefore, to get consistent and reliable results, it was necessary to perform some changes to the non-stationary data to get a more statistically stable environment[57].

A simple measure to make the seasonal data more stationary was to differentiate it, and an illustration of this point is given in fig. 4.2. Here, the function $f(x) = \sin x + x$ is clearly non-stationary and will continue to grow with a growing x. However, its derivative, $f'(x) = \cos x + 1$ is stationary. This also applies to the seasonal trends in the time series data, where the temperature will depend on the time of the year. Differentiating the data was likely to remove some of this seasonal non-stationarity. It also aided in bridging the gap between the test and training sets, as the temperature's derivative is likely less seasonally dependent than the temperature itself.

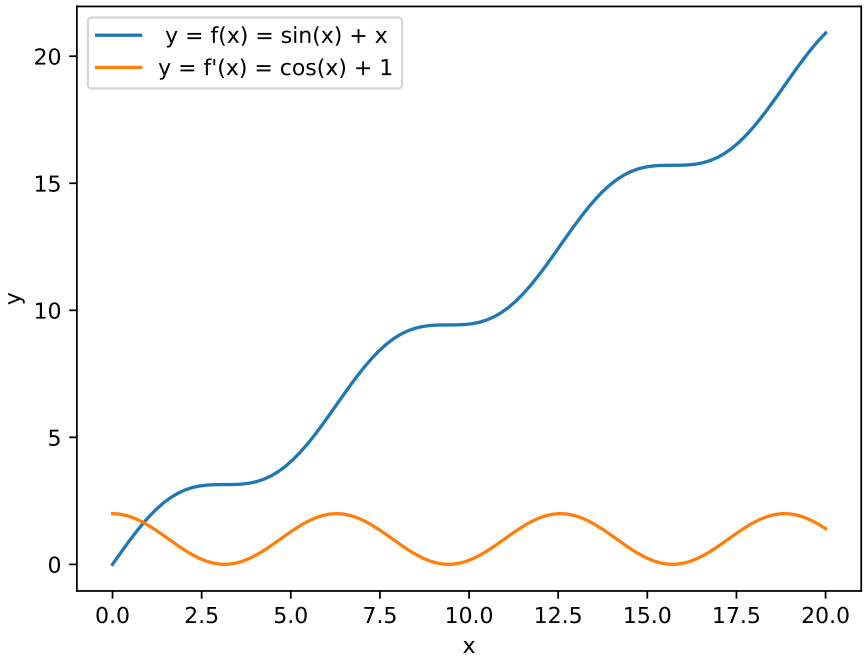


Figure 4.2.: Depiction of the non-stationarity of $f(x) = \sin x + x$ in contrast to its derivative

Subsequently, the data were normalized. Normalization is essential both for computational and accuracy purposes. Z-score normalization, described by eq. (2.4), was deemed a good choice for normalizing the data in hand due to its more efficient handling of potential outliers. The z-score normalizer sklearn.preprocessing.StandardScaler from sklearn was used to perform the normalization[58]. The normalizer was first fitted on the entire train and validation set to determine the values of μ_i and σ_i, i.e., the mean and standard deviation of the respective features. The normalizer was not fitted on the test data to avoid any data leakage this would introduce, which is deemed a good and vital practice. Subsequently, the train, validation, and test data were normalized using the determined values of μ_i and σ_i from the fitting.

After this, the data windowing was performed. When performing data windowing, two main parameters have to be decided upon: feature sequence length and label sequence length. How long the look-back period, i.e., feature sequence length, and the forecasting horizon, i.e., label sequence length, should be is not necessarily evident. Regarding the look-back period, a longer period enables the model to capture longer-term dependencies at the expense of potentially diluting the short-term dependencies[33]. Regarding the forecasting horizon, a too-long horizon may cause the training loop to fail to converge,

while a too-short horizon may cause the rolling forecast to diverge, from the author's experience. Through trial and error, the look-back period was set to 96 timesteps, i.e., one day of measurements, while the forecasting horizon was set to eight timesteps, i.e., two hours.

With the data windowing in place, the data preprocessing procedure was done. The following sections will first introduce the door model implemented to facilitate a rolling forecast. After that, the two DDM architectures employed to perform the forecasting task and their architectures and hyperparameters will be presented.

4.2.2. Door Model

To facilitate a rolling forecast, it was necessary to have a model predicting whether the doors were open or closed. The forecasting model requires the state of the doors and temperatures as input. Hence, to make it possible to forecast further into the future than the forecasting horizon of the model, there was a need for a model providing a prediction of the door states. The simplest way to achieve this would be to forward-fill these values from the last observed door states. However, this would be a significant simplification and would likely deteriorate the models' performance.

Therefore, a simple FFNN with two hidden layers was implemented and trained to make this prediction based on the time of day and temperatures in each room. After training and tuning its hyperparameters, this model achieved an accuracy of 83% on the test set, which is deemed a good indicator of whether the doors are open or closed. In addition, the same training, validation, and test split described previously was used to avoid data leakage for this model. Table 4.1 summarizes the hyperparameter search.

Hyperparameter	**Search Range**	**Best Hyperparameter**
Batch Size	500-3000	2000
Epochs	Early Stopping	37
Learning Rate	$10^{-1} - 10^{-5}$	10^{-4}
L1 Regularization	$10^{-2} - 10^{-5}$	10^{-3}
Number of FFNN Hidden Layers	1 - 4	2
Width of FFNN Hidden Layers	32-256	64

Table 4.1.: Door model: Hyperparameter search

4.2.3. LSTM

Model Architecture

The LSTM architecture described in section 2.3.3 was implemented using the PyTorch framework[52]. The implemented architecture is depicted in fig. 4.3. This architecture expands on the vanilla LSTM architecture by incorporating several hidden layers in the FFNN after the LSTM, which was found to provide more satisfying results.

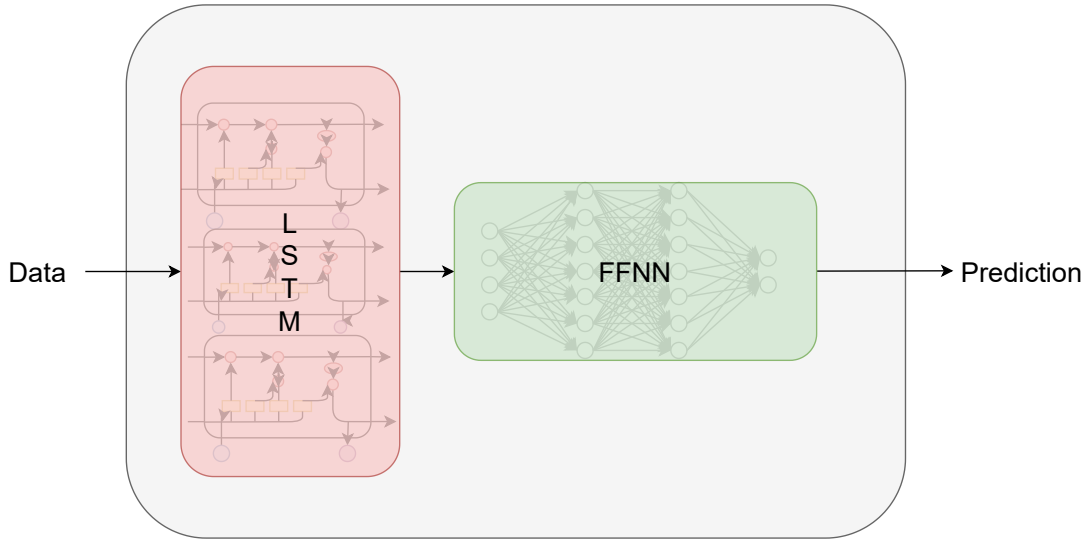


Figure 4.3.: Architecture of the implemented LSTM model

During training and inference, the input data were first fed into the LSTM layer. The output from the LSTM layer was then fed into a multilayer FFNN, responsible for interpreting and sorting the output from the LSTM layer to their corresponding forecasting variable.

Training Routine

The training routine implemented for this model followed a standard routine for training a neural network with time-series data, using the PyTorch framework[52]. The model first tried to forecast an input of dimension $(batch\_size, 96, 22)$ where 96 represents the look-back period, and 22 represents the number of features. This gave an output of dimension $(batch\_size, 8, 10)$ where 8 represents the forecasting horizon of eight timesteps, and 10 represents the number of forecasted variables.

These outputs were then compared to their corresponding label using a mean squared error (MSE) loss function before the gradients of the loss function with respect to the network weights were calculated. The Adam algorithm was used to calculate the updated weights, and the weights were updated accordingly for each batch in the training set. The weights of the network were initialized according to the He initialization scheme.

After the whole training set had been iterated, its performance was examined on the validation set. This procedure was implemented in a more sophisticated way than a vanilla validation loop. Instead of simply comparing the 8-timestep output of the model with its label, the validation loop utilized a rolling forecast of 96 timesteps to examine the model's accuracy, as previously introduced in pseudo-code in algorithm 1. Each window in the validation set was forecasted with a rolling forecast of $n\_iterations = 12$ and $forecast\_horizon\_length = 8$, providing a forecast of 96 timesteps per validation window. This rolling forecast was then compared to the actual evolution of temperature on a mean average error (MAE) basis for assessing the model's performance on unseen data.

The validation MAE was used to implement early stopping, which is an effective measure to prevent overfitting, and also eliminates the need for tuning the number of training epochs, as the training automatically halts once the validation loss starts to increase. Since the validation loss was calculated on the same rolling forecast basis as the model was intended to be used in a real-life scenario, this was deemed a good metric for assessing the model's performance. The patience of the early stopping was set to 10.

Hyperparameters

Such a model has several essential hyperparameters, many of which impact each other, and tuning the hyperparameters is, therefore, inherently complex. As a result, numerous papers have been written with sophisticated ways of doing hyperparameter search[59]. However, for the purpose and scope of this study, one of the simplest types of hyperparameter search was utilized, namely grid search.

The grid search started with a broad search before the grid was condensed around the areas that showed the most promise. Table 4.2 summarizes the conducted grid search for each hyperparameter, along with its corresponding best-discovered hyperparameter. It is worth noting that the batch size parameter is a parameter that advantageously should have been tested with even higher values. However, this was not possible due to computational limitations on the available equipment.

Hyperparameter	**Search Range**	**Best Hyperparameter**
Batch Size	50-1000	1000
Epochs	Early Stopping	6
Learning Rate	$10^{-1} - 10^{-5}$	10^{-2}
L1 Regularization	$10^{-2} - 10^{-15}$	10^{-12}
Number of LSTM Layers	1 - 2	1
Hidden LSTM Size	32 - 256	64
Number of FFNN Hidden Layers	1 - 5	3
Width of FFNN Hidden Layers	32-256	64

Table 4.2.: LSTM: Hyperparameter search

4.2.4. Transformer

Model Architecture

As the Transformer architecture proposed by Vaswani et al., depicted in fig. 2.6a, is designed to operate on NLP problems, some modifications had to be made to the model to make it suitable for time-series forecasting. First, the final softmax layer, responsible for transforming outputs into output probabilities, had to be removed.

Additionally, the input and output embeddings had to be modified. Since the time-series data is already understandable for the model, i.e., real-valued unlike words in an NLP problem, these embeddings can be substituted with a linear layer. These layers convert their inputs from the n-dimensional input space into the same dimension as the

model. The implemented model is depicted in fig. 4.4.

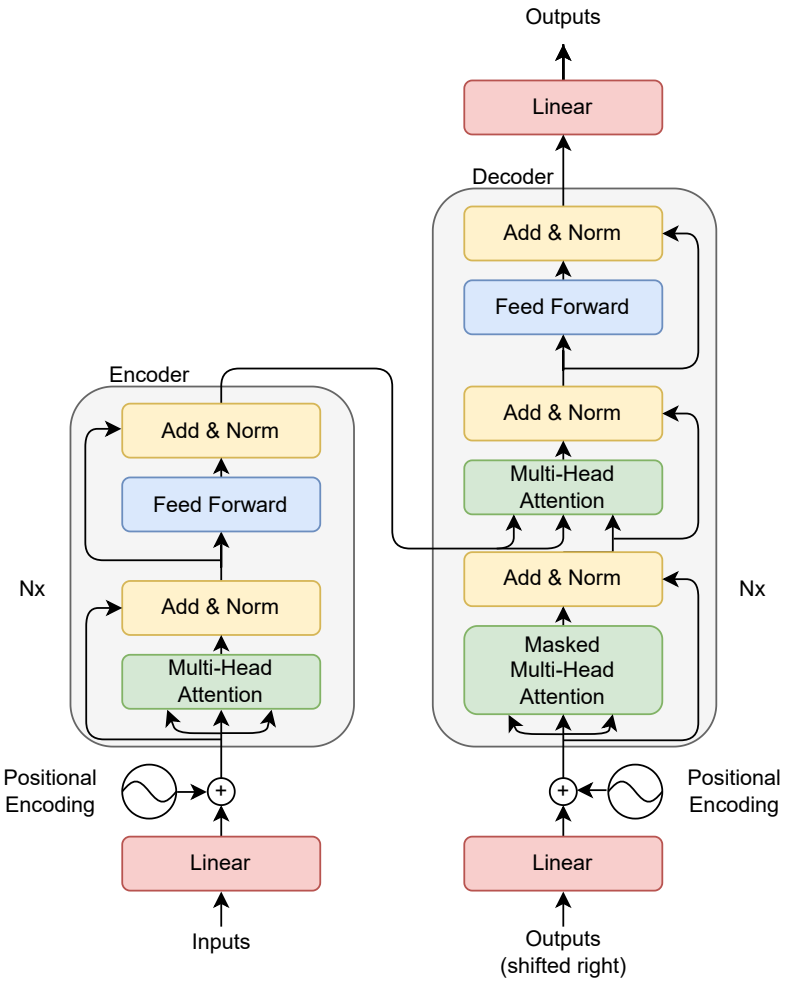


Figure 4.4.: Architecture of the implemented Transformer model, inspired by Vaswani et al.[46]

The positional encoding was implemented in the same way as described in section 3.2.2, with handcrafted daily, weekly, and yearly frequencies. The main difference between the LSTM and Transformer input is that for the Transformer, this positional encoding was added to the input after the linear layer and not directly added to the feature space as with the LSTM.

This model was also implemented using the PyTorch framework for the Transformer architecture[52].

Training Routine

The training routine followed the same procedure as the LSTM, adjusted with the specific Transformer training procedure described in section 2.3.4. In addition, the validation error was also evaluated with the same rolling forecast as the LSTM, with patience of 10 for early stopping.

Hyperparameters

The hyperparameter tuning of a Transformer is a complicated process. A Transformer has several internal hyperparameters that will affect its capability and ability to learn, such as the dimension of the model itself, the number of encode layers, and the number of decode layers, to name a few. Even though it also here exist sophisticated ways of pursuing the optimal hyperparameters, the scope of this study limited the hyperparameter tuning to a grid search.

The range of search for the different hyperparameters is presented in table 4.3, together with the corresponding best-discovered hyperparameters. It is also worth noting that an increased batch size was impossible due to computational limits on the available equipment, but it would be interesting to examine.

Another point worth mentioning regarding grid search in general, and especially with as many hyperparameters as a Transformer introduce, is that there most likely exists more optimal hyperparameters. However, the conducted grid search is expected to have brought the hyperparameters reasonably close to the optimal values.

Hyperparameter	**Search Range**	**Best Hyperparameter**
Batch Size	50 - 500	500
Epochs	Early Stopping	3
Learning Rate	10^{-2} - 10^{-5}	10^{-3}
L1 Regularization	10^{-3} - 0	10^{-15}
Dropout Probability	0 - 0.5	0.3
Dimension of Model	32 - 512	128
Dimension of FF-layers	32 - 2048	256
Number of Heads	4 - 8	8
Number of Encode Layers	2 - 6	4
Number of Decode Layers	2 - 6	4

Table 4.3.: Transformer: Hyperparameter search

4.2.5. Scenario and Simulation Setup

As previously mentioned, the simulation was set up to simulate the month of July 2022, and this month was therefore held aside in the test set. In addition, the simulation was set up to support an arbitrary length on the rolling forecast, providing the opportunity to modify and examine how the correction interval impacted the model's accuracy. In this context, the correction interval is defined as the duration for which the model makes forecasts into the future before getting corrected with the true temperatures.

With the door model predicting the states of the doors, the LSTM/Transformer forecasting the indoor temperature, and the future temporal features being inherently known,

the only state not predicted, forecasted, or known, was the evolution of the outdoor temperature. The initial plan was to use weather forecasts as input to the model during the rolling forecasts. According to Anders Doksæter Sivle from the Norwegian Meteorological Institute and Anna Kathinka Dalland Evans from the University of Oslo, Norwegian weather forecasts typically achieve 70% - 80% accuracy for temperature forecasts over the next few days[60]. This level of accuracy was deemed a good indicator of how the outdoor temperature would evolve.

However, retrieving historical data on Norwegian weather forecasts proved to be challenging, if not impossible, and according to the author's best knowledge and exploration, such data is impossible to acquire. Nevertheless, according to Sivle and Evans' statement regarding weather forecast temperature accuracy, an emulation of a weather forecast can be reconstructed from the actual outdoor temperature measurements.

This reconstruction was achieved by adding a random bias term in the range of ±30% to the actual measured temperatures. The bias term was updated to a new random number every two hours, as it is reasonable to assume that the weather forecast has more of a constant offset for periods than jumping around the actual temperature every 15 minutes. Even though this "weather forecast" is not perfect, it is assumed to be a good indicator of the actual weather forecast and that it will not impact the forecast too significantly. The models were also double-checked with "weather forecasts" significantly poorer to ensure they did not rely too significantly on the forecasted outdoor temperature.

4.3. Hybrid Analysis and Modeling - HAM

A HAM model, more specifically a CoSTA model, was also developed for this specific forecasting task. Implementing a CoSTA model is more comprehensive and requires more domain knowledge than DDMs.

At first, a PBM of the asset was developed as the cornerstone of the CoSTA. After that, the DDM was developed to learn the corrective source term. This section introduces the method used in this work.

In this section, fig. 4.5 and table 4.4 is included from the specialization project leading up to the thesis for completeness. The values in tables 4.5 and 4.6 were derived based on values obtained from the same specialization project[1].

4.3.1. Data Preprocessing

For this model, a new feature was added to the dataset, namely the Global Horizontal Irradiance (GHI). GHI is defined as the sum of the shortwave direct and diffuse radiation from the sun on a horizontal surface measured in the unit W/m^2, and is highly influential regarding the thermophysics of a building due to the heating gain it may pose. This data was obtained from the Norwegian Centre for Climate Services (NCCS)[61]. NCCS records the average GHI every hour at a climate station located at Gløshaugen, Trondheim, just over 5km from the asset. These measurements are therefore believed to

be a good indicator of the radiance at the asset.

The train, validation, and test split was kept the same as the previous models. However, the data preprocessing when dealing with a CoSTA has to be done differently since the data can not be normalized in the same way as with a pure DDM. This is because the PBM part of the CoSTA operates on inputs in the unit of °C, and normalizing the input to the PBM would not make any sense. However, the input to the DDM should be normalized to achieve the best performance.

Given that the PBM requires the original data as input, while the DDM benefits from normalized data as input, the dataset was divided into two copies. One of the copies kept the actual measurements to be used with the PBM. For the second copy, the continuous variables were first time-differentiated for the same reasons described in section 4.2.1 before the data was z-score normalized for additional input to the DDM.

These dataset copies were then windowed together, providing each data window with inputs for the PBM, DDM, and corresponding labels.

4.3.2. Door Model

The same door model utilized for the DDMs, as detailed in section 4.2.2, was also employed to facilitate the rolling forecast for the CoSTA.

4.3.3. Model Architecture

PBM

The PBM part of the CoSTA was implemented as the RC network introduced in section 2.2.1, described by eq. (2.2), with some modifications, which will be described in the following paragraphs.

First, the Q-values describing the energy flux from the sun on the different nodes were simplified. These values are challenging to derive accurately, as the radiation is defined as W/m^2 on a horizontal surface and also depends on the cardinal directions of the walls. Therefore, the total radiation was subdivided into different rooms based on the area of the room. This subdivision is naturally not remarkably accurate, but the plan was to let the DDM part of the CoSTA handle the more exact distribution of the radiation. The DDM should benefit from having access to the radiation data and the daily fluctuations the radiation poses to the PBM.

Secondly, the heating loads for each zone were set to a constant of 10 watts to emulate radiators and heating from occupants, with the cooling load set to 0 watts. This constant gain is also a significant simplification, but the asset does not have measurements on any of these values, and these values will therefore become speculative. The plan was consequently to let the DDM part also model this discrepancy.

The implemented asset, resembling the actual dimensions and construction of the physical asset, is graphically depicted in fig. 4.5. For a more detailed floor and section plan, appendices A.1.1 and A.1.2 can be examined. The color codes of fig. 4.5 is listed in table 4.4. Figure 4.5 and table 4.4 were extracted from the specialization project leading up to this thesis[1].

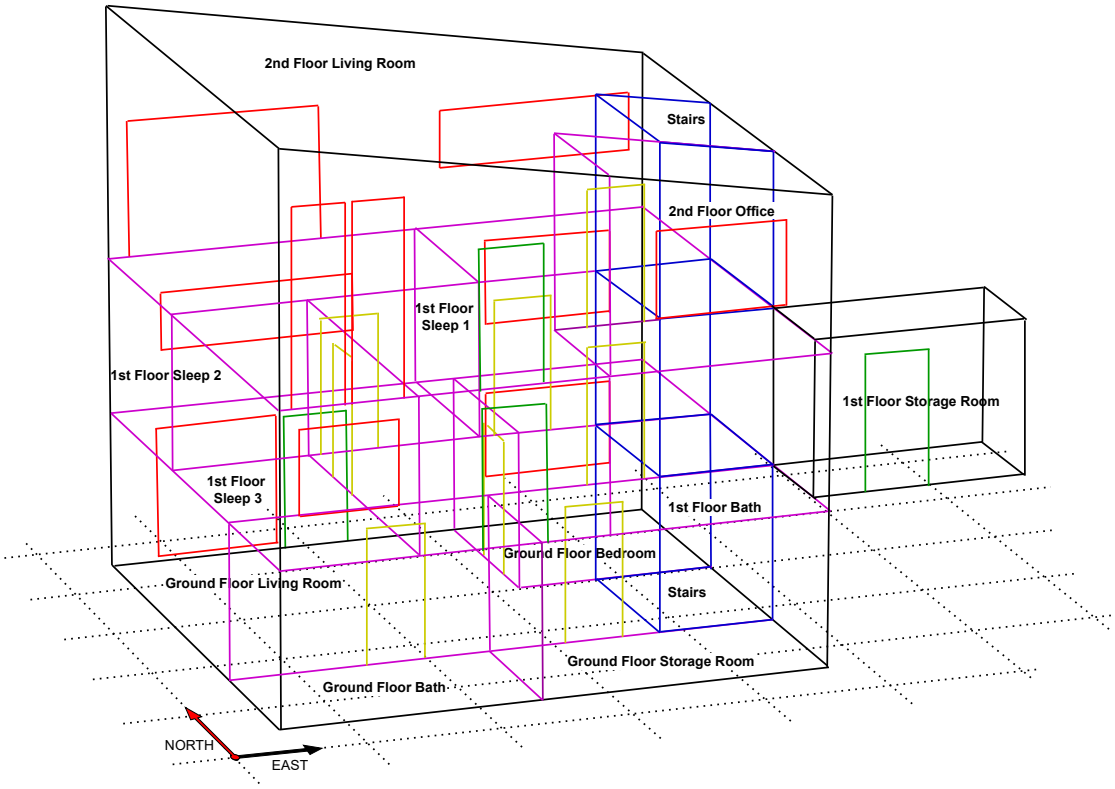


Figure 4.5.: Depiction of the asset integrated into CoSTA

Color	Explanation
Black	Frame of the asset
Red	Window
Green	Exterior door
Yellow	Interior door
Blue	Stairwell
Purple	Partition wall

Table 4.4.: CoSTA-PBM: Color-coding of fig. 4.5

The asset was implemented according to the RC network, with the following coefficient values for each zone.

	R_{ext} [W/K]	$R_{outWall}$ [W/K]	R_{inWall} [W/K]	R_{room} [W/K]	C_{wall} [J/K]	C_{room} [J/K]
Ground Floor Bedroom	0.341	1.366	1.366	0.341	7873	25933
Ground Floor Living Room	0.598	2.392	2.392	0.598	19862	49387
Stairs	0.407	1.627	1.627	0.407	8319	37232
Ground Floor Bathroom	0.271	1.082	1.082	0.271	7057	26584
Ground Floor Storage Room	0.290	1.161	1.161	0.290	7046	27892
1st Floor Bedroom 1	0.367	1.468	1.468	0.367	11713	31406

Continued on next page

	R_{ext} [W/K]	$R_{outWall}$ [W/K]	R_{inWall} [W/K]	R_{room} [W/K]	C_{wall} [J/K]	C_{room} [J/K]
1st Floor Bedroom 2	0.421	1.685	1.685	0.421	15780	38716
1st Floor Bedroom 3	0.371	1.483	1.483	0.371	10850	32691
1st Floor Bathroom	0.173	0.691	0.691	0.173	9847	21833
1st Floor Storage Room	0.424	1.695	1.695	0.424	2751	35144
1st Floor Entrance Hall	0.123	0.493	0.493	0.123	19392	24861
2nd Floor Living Room	2.624	10.497	10.497	2.624	35695	27910
2nd Floor Office	0.381	1.522	1.522	0.381	8110	29130

Table 4.5.: CoSTA-PBM: Zone coefficients

The following connections between the zones were implemented.

Connected zones		$R_{partWall}$ [W/K]	$C_{partWall}$ [J/K]
Ground Floor Bedroom	Ground Floor Living Room	10.352	8968
Ground Floor Bedroom	Stairs	4.437	3843
Ground Floor Bedroom	1st Floor Bedroom 1	2.016	10350
Ground Floor Living Room	Ground Floor Bathroom	9.785	8477
Ground Floor Living Room	Stairs	38.109	0
Ground Floor Living Room	1st Floor Entrance Hall	2.722	13973
Ground Floor Living Room	1st Floor Bedroom 2	3.226	16561
Ground Floor Bathroom	Ground Floor Storage Room	5.916	5124
Ground Floor Bathroom	1st Floor Bedroom 3	1.129	5796
Ground Floor Storage Room	Stairs	4.437	3843
Ground Floor Storage Room	1st Floor Bathroom	1.828	9384
1st Floor Bedroom 1	1st Floor Bedroom 2	7.394	6406
1st Floor Bedroom 1	1st Floor Entrance Hall	4.437	3843
1st Floor Bedroom 1	Stairs	4.437	3843
1st Floor Bedroom 1	2nd Floor Living Room	2.419	12420
1st Floor Bedroom 2	1st Floor Bedroom 3	5.176	4484
1st Floor Bedroom 2	1st Floor Entrance Hall	6.655	5765
1st Floor Bedroom 2	2nd Floor Living Room	3.226	16561
1st Floor Bedroom 3	1st Floor Entrance Hall	12.570	10890
1st Floor Bedroom 3	2nd Floor Living Room	2.399	12317
1st Floor Entrance Hall	1st Floor Bathroom	3.944	3416
1st Floor Entrance Hall	Stairs	38.109	0
1st Floor Entrance Hall	2nd Floor Living Room	4.281	21977
1st Floor Bathroom	1st Floor Storage Room	0.722	5800
1st Floor Bathroom	Stairs	4.437	3843
1st Floor Bathroom	2nd Floor Office	1.828	9384
2nd Floor Living Room	2nd Floor Office	12.090	10474
2nd Floor Living Room	Stairs	43.131	0
2nd Floor Office	Stairs	4.729	4097

Table 4.6.: CoSTA-PBM: Zone connections

All of the values were derived by inspecting the U-value and heat capacity of the cor-

responding surfaces implemented in ESP-r during the specialization project[1]. The U-values were based on imposed Norwegian standards from The Regulations on Technical Requirements for Construction Works, which impose strict requirements for U-values in newer buildings[62]. The heat capacities were derived from typical values in the construction industry. For further details on the gathering of this data, the specialization report can be reviewed[1].

In order to solve the system, the ODE in eq. (2.2) was approximated by the explicit Euler method in the following manner.

$$\vec{T}^{n+1} = \vec{T}^{n} + h \cdot \dot{\vec{T}}(t^n) \tag{4.1}$$

where

$$\dot{\vec{T}}(t^n) = \begin{pmatrix} \dot{\vec{T}}_{\text{room}}(t^n) \\ \dot{\vec{T}}_{\text{wall}}(t^n) \end{pmatrix} = \begin{pmatrix} C_1 & 0 \\ 0 & C_2 \end{pmatrix}^{-1} \cdot \begin{pmatrix} D & E \\ F & G \end{pmatrix} \cdot \begin{pmatrix} \vec{T}_{\text{room}}(t^n) \\ \vec{T}_{\text{wall}}(t^n) \end{pmatrix} + \begin{pmatrix} \vec{u}_{\text{room}}(t^n) \\ \vec{u}_{\text{wall}}(t^n) \end{pmatrix}. \tag{4.2}$$

C_1 and C_2 are diagonal matrices with solely positive values on the diagonal and will always be invertible. As this is a relatively slow system, the step length was set to 60 seconds. Therefore, the system had to be solved iteratively 15 times to reach the subsequent sampled sensor measurement, which has a sampling period of 15 minutes.

DDM

For this work, the DDM was implemented as a combination of an LSTM and an FFNN in order to be able to catch more of the long-term (and short-term) dependencies than a pure FFNN. This model combined the CoSTA model described in section 2.4 and the LSTM model implemented in section 4.2.3 in the way depicted in fig. 4.6. Additionally, a z-score normalizer was implemented between the PBM output and FFNN input to ensure that the input data to the FFNN was normalized. This normalizer's μ and σ coefficients were decided according to the forecasted temperatures produced by the PBM without corrective source term on the training set.

This proposed architecture expands on the vanilla CoSTA architecture proposed by Blakseth et al.[12] by incorporating an extra LSTM layer and providing it with the previous N timesteps The z-score normalization after the PBM is also an expansion of the original architecture. The rationale behind this implementation is that the LSTM layer should be able to catch more of the time dependencies which the plain FFNN would be struggling with. In addition, the source term for consecutive timesteps will most likely be correlated; for example, if there is a fire in the fireplace or there are more people than usual in a room, this will be reflected in consecutive source terms, and the LSTM layer should be able to model these dependencies better than a plain FFNN. As the output from the LSTM will be between -1 and 1, it is also believed that normalizing the output from the PBM will benefit the model.

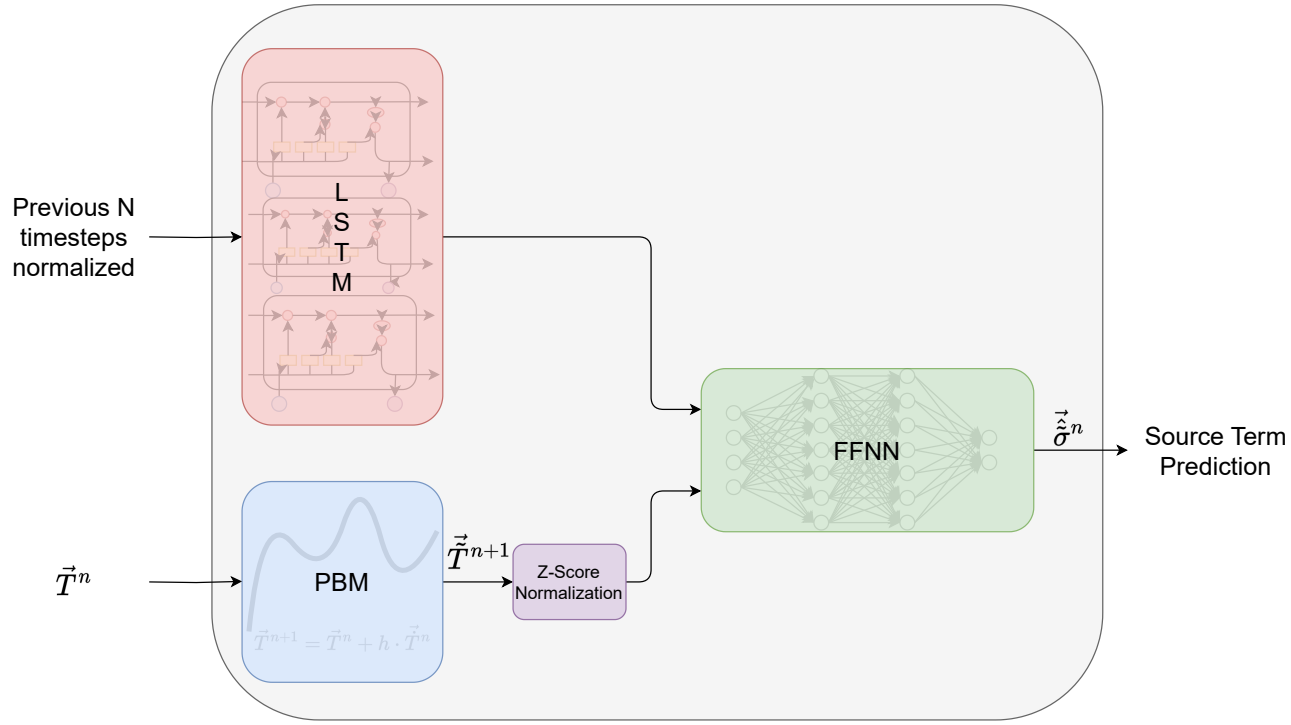


Figure 4.6.: Architecture of the implemented CoSTA DDM component

4.3.4. Training Routine

The training routine for this model followed the standard training procedure of a CoSTA, as described in section 2.4.

The fact that the utilized dataset, based on the sensor measurements from the asset, only has measurements from seven of the 13 zones implemented in the PBM introduced some difficulties during the labeling process. It was found necessary to implement all 13 rooms in the PBM to make better-educated guesses of the correct resistances and capacitances of the model, even though some of the implemented rooms would not have labels.

This was solved by only labeling the rooms that had true temperature measurements in order to calculate the error gradients. Given that the dynamics in the source term of the different rooms likely resemble each other to some extent, the provided labels should be able to describe the major part of the source term in all of the rooms. Despite only a subset of the zones in the PBM being labeled, the DDM should, therefore, still be capable of modeling the error in the underlying equations. By applying regularization techniques such as L1 regularization and dropout, the DDM was expected to be able to generalize the corrective source term for the different rooms. However, this is not optimal and likely complicates the learning process.

The implemented model used both L1 regularization and dropout in the FFNN layers to prevent overfitting and inappropriate co-adaptations between neurons in these layers in order to facilitate better generalization in predicting the corrective source term.

4.3.5. Hyperparameters

The hyperparameter tuning of the CoSTA followed the same procedure as the LSTM described in section 4.2.3 due to the notable similarities between the DDM part of the CoSTA and the implemented LSTM. Although experimenting with a larger batch size could potentially benefit the model, this was not possible due to the computational complexity of the implemented CoSTA and the limitations of the available equipment. The following table summarizes the hyperparameter tuning.

Hyperparameter	**Search Range**	**Best Hyperparameter**
Batch Size	50-100	100
Epochs	Early Stopping	4
Learning Rate	$10^{-1} - 10^{-5}$	10^{-3}
L1 Regularization	$10^{-2} - 10^{-15}$	10^{-3}
Dropout Probability	$0 - 0.5$	0.4
Number of LSTM Layers	1 - 2	1
Hidden LSTM Size	32 - 256	64
Number of FFNN Hidden Layers	1 - 5	3
Width of FFNN Hidden Layers	32-256	64

Table 4.7.: CoSTA-DDM: Hyperparameter search

4.3.6. Scenario and Simulation Setup

The scenario and simulation setup for the CoSTA followed the same procedure as described for the DDMs in section 4.2.3. Firstly, the pure PBM was evaluated to examine its accuracy and determine whether the forecasts from the PBM were reliable enough for the CoSTA. Subsequently, the complete CoSTA setup was examined.

5. Results and Discussions

This chapter presents and discusses the resulting forecasts from the proposed models described in chapter 4, and compares them to actual measurements from the asset's temperature sensors. The results are assessed using Mean Absolute Error (MAE) and visual inspection of the forecasts plotted against the actual measurements, and the deviations between the forecasted and the actual measured temperatures are discussed. Finally, any lessons learned from this work are assessed.

5.1. Physics-Based Modeling - PBM

This section presents the resulting forecast from the PBM part of the CoSTA and compares it to the forecast produced by the more thorough PBM developed in the framework ESP-r during the specialization project[1]. This thesis only briefly discusses the results from the PBM developed in the specialization project. For a more comprehensive discussion, the specialization report should be reviewed. The forecasts from the PBM component of CoSTA are examined to assess their suitability as a solid foundation for further CoSTA integration.

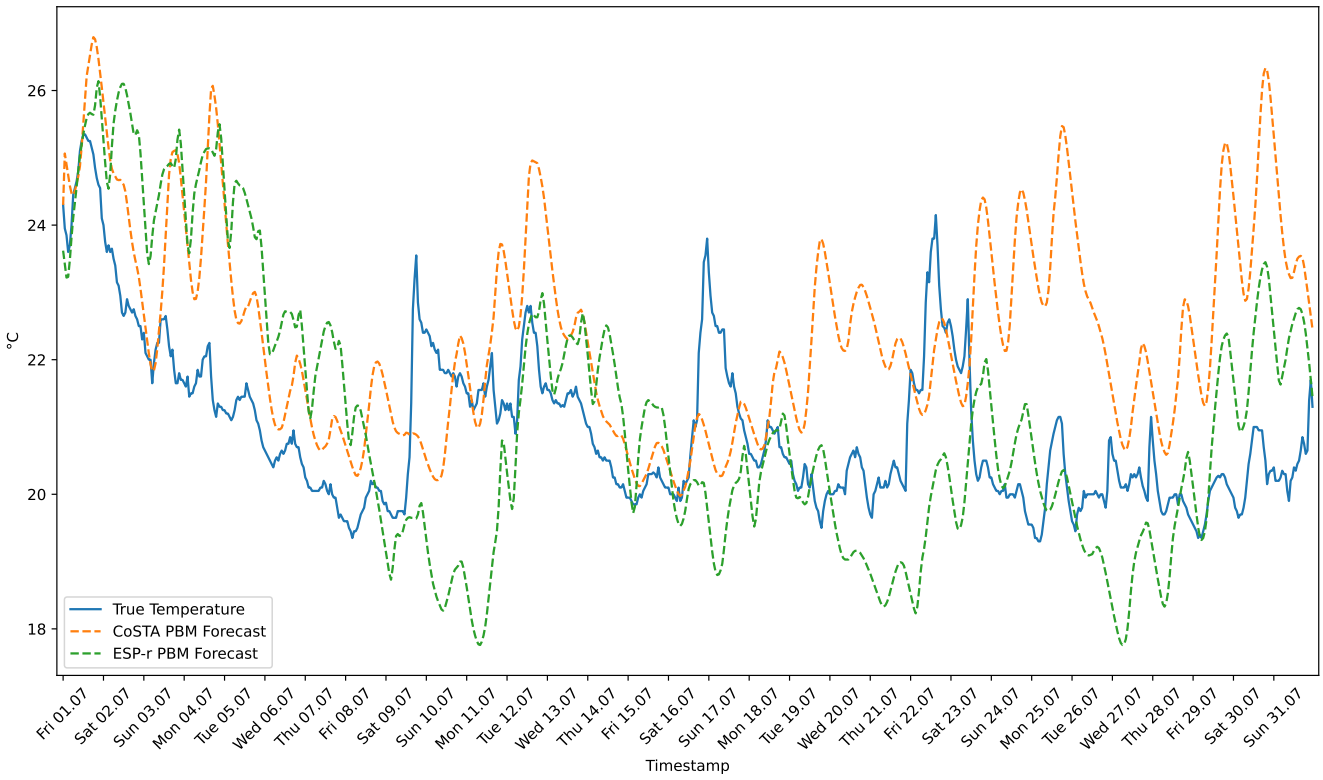
5.1.1. Results

Table 5.1 describes the forecast MAE of the different models, and the forecasts produced are depicted in fig. 5.1. Note that the 2nd floor kitchen is not included in these models because the kitchen is part of an open solution with the living room, and determining the conduction and heat capacity of the separating "wall" therefore becomes difficult for the PBMs. Hence, the kitchen is incorporated into the 2nd floor living room.

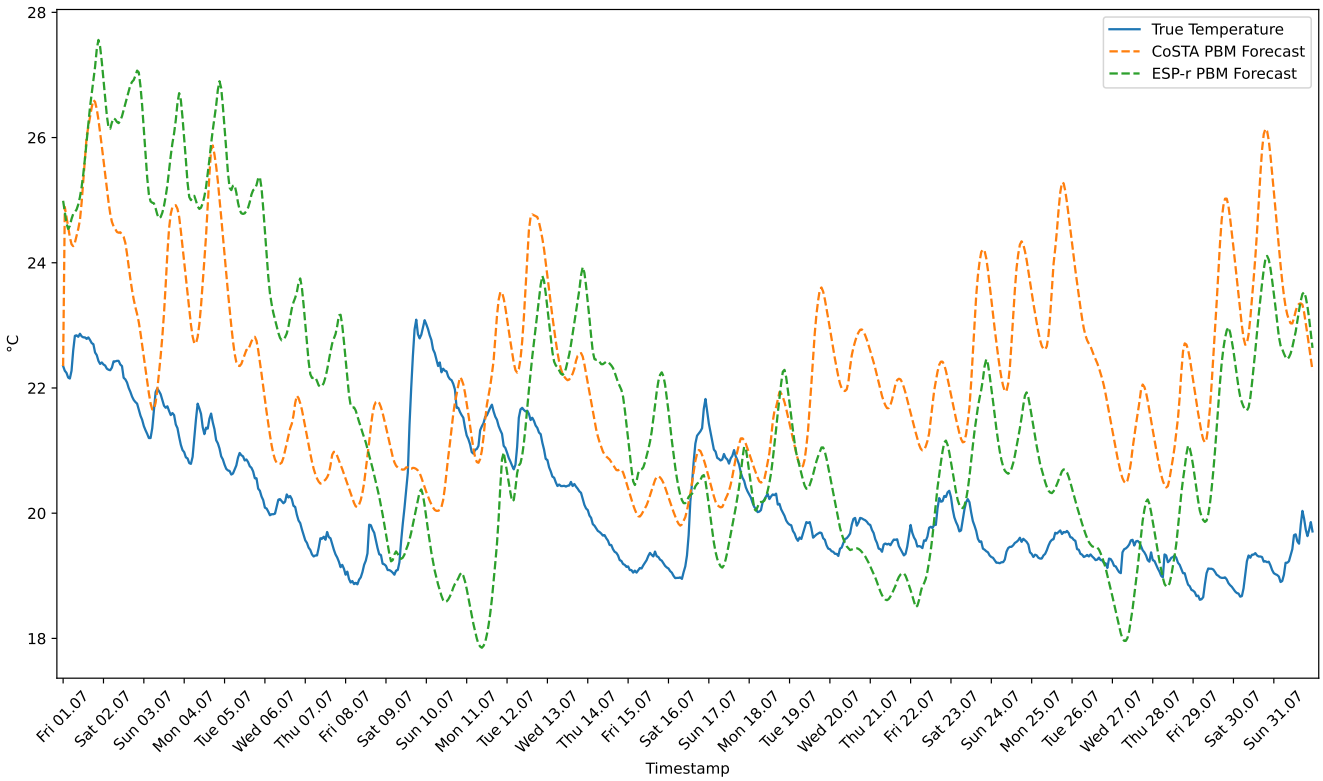
The outdoor temperature plotted in fig. 5.2 will serve as a reference and an explanation for the PBM forecasts.

	Mean Absolute Error[°C]	
Room	**ESP-r PBM**	**CoSTA PBM**
Ground Floor Bedroom	1.53	1.74
Ground Floor Living Room	2.11	2.26
1st Floor Entrance Hall	1.99	1.36
1st Floor Bedroom 1	1.53	1.18
1st Floor Bedroom 2	1.98	1.50
2nd Floor Living Room	2.41	1.91
2nd Floor Office	4.19	1.98
Average MAE	**2.25**	**1.71**

Table 5.1.: PBM: Mean absolute error of the different rooms

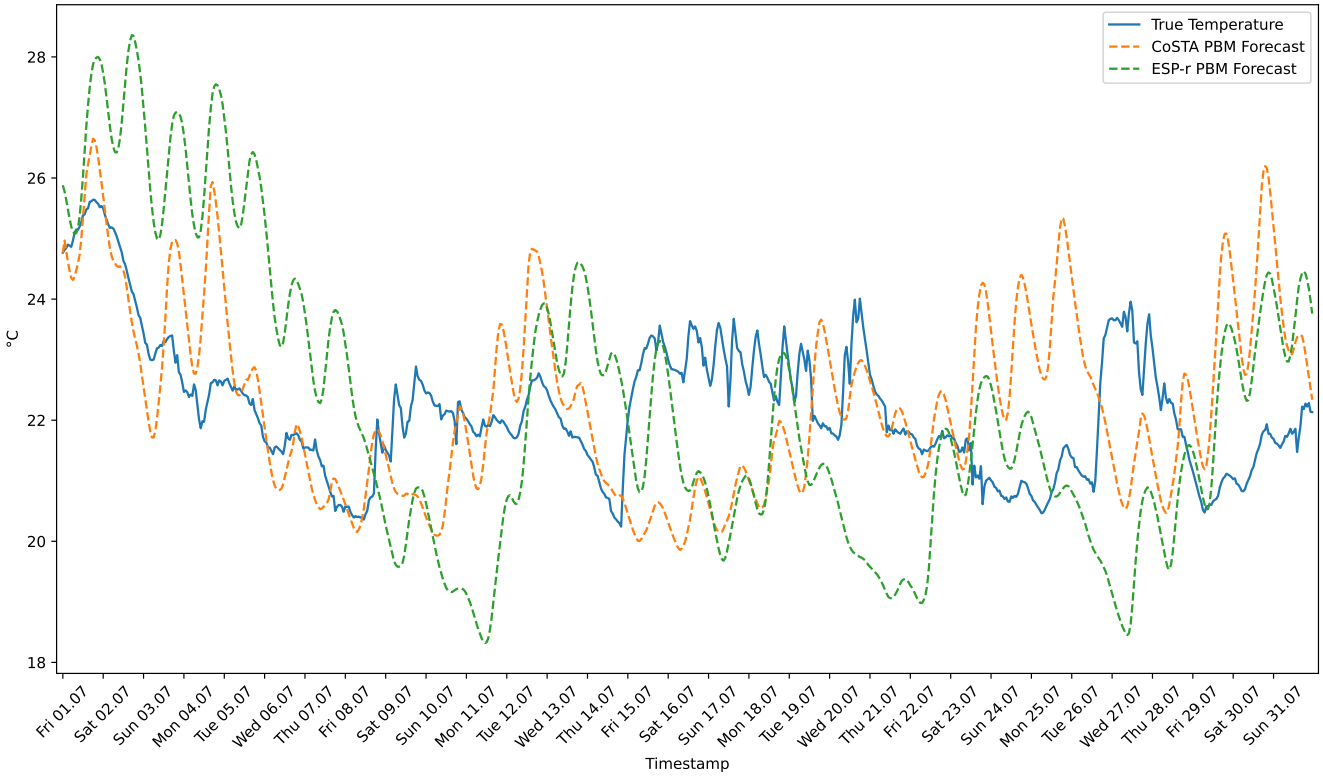


(a) Ground floor bedroom forecasts

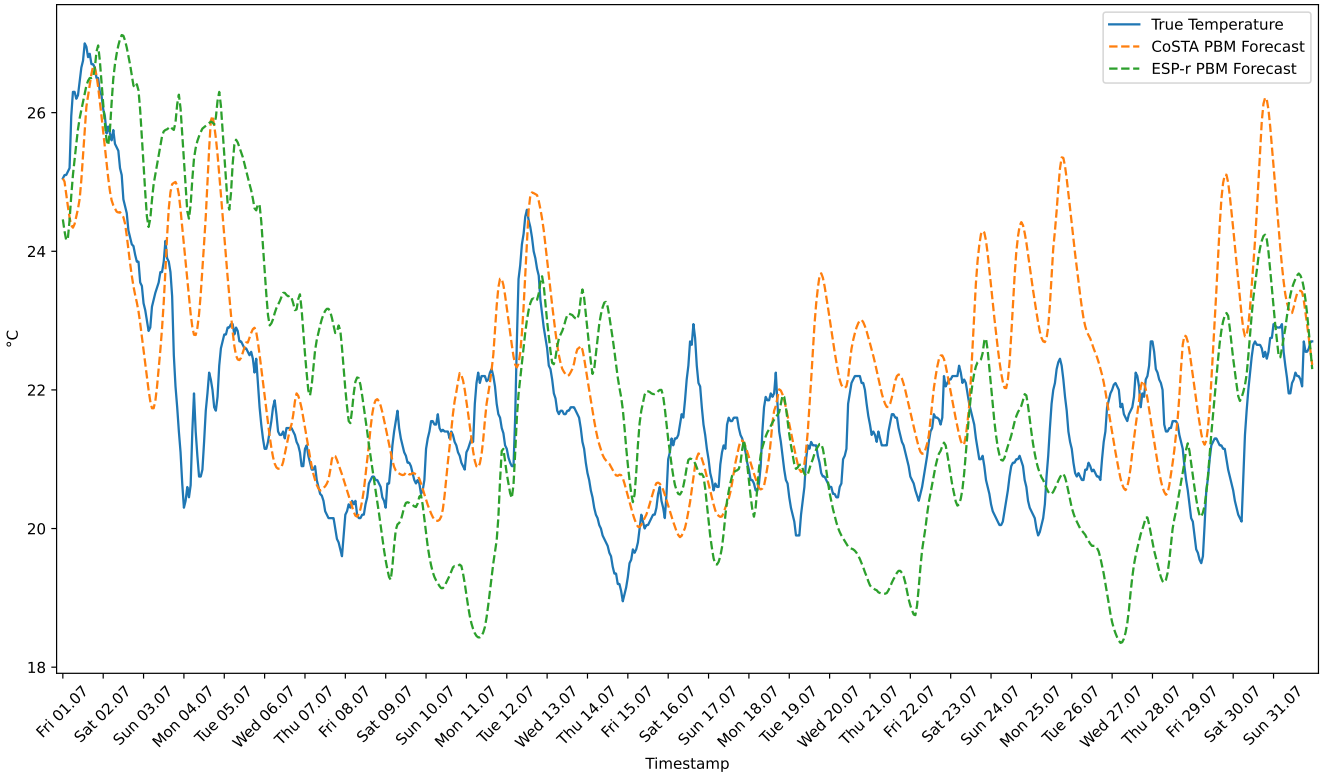


(b) Ground floor living room forecasts

Figure 5.1.: PBM: True and forecasted temperatures of different rooms

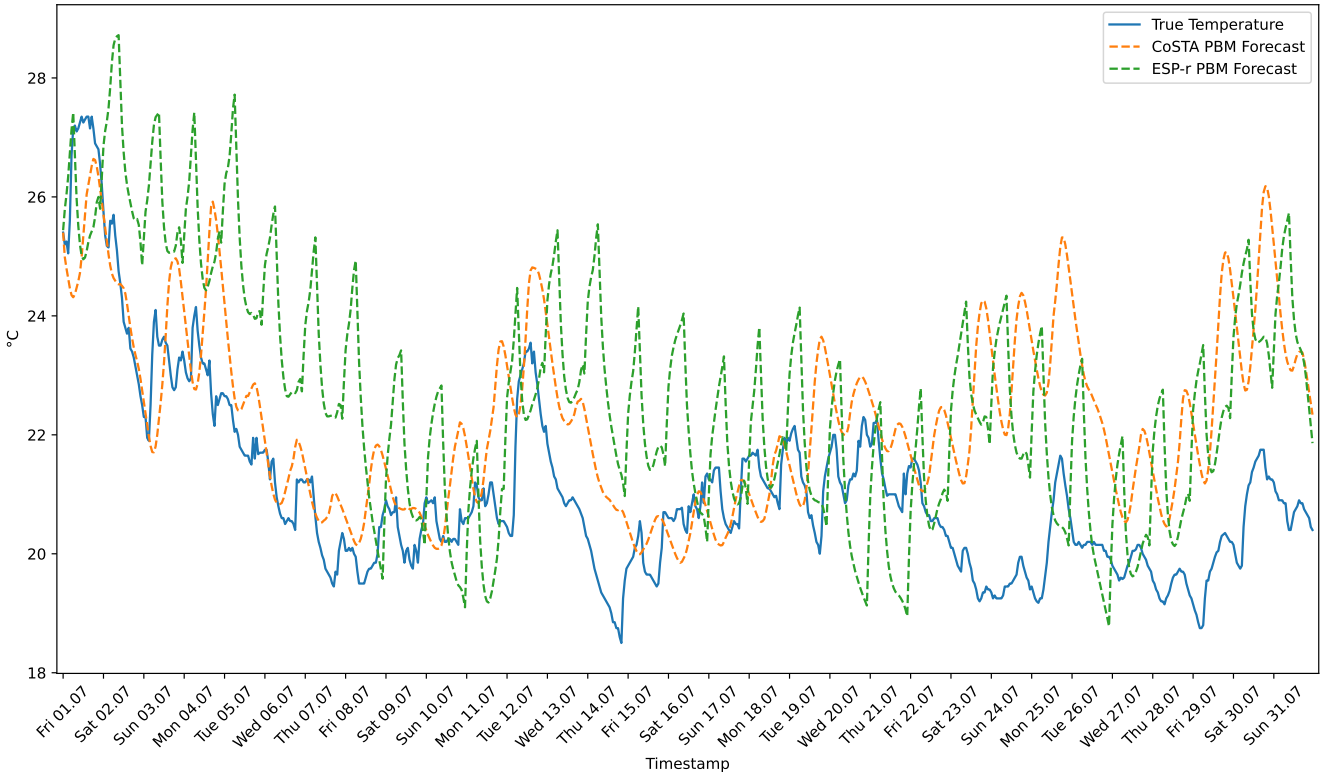


(c) 1st floor entrance hall forecasts

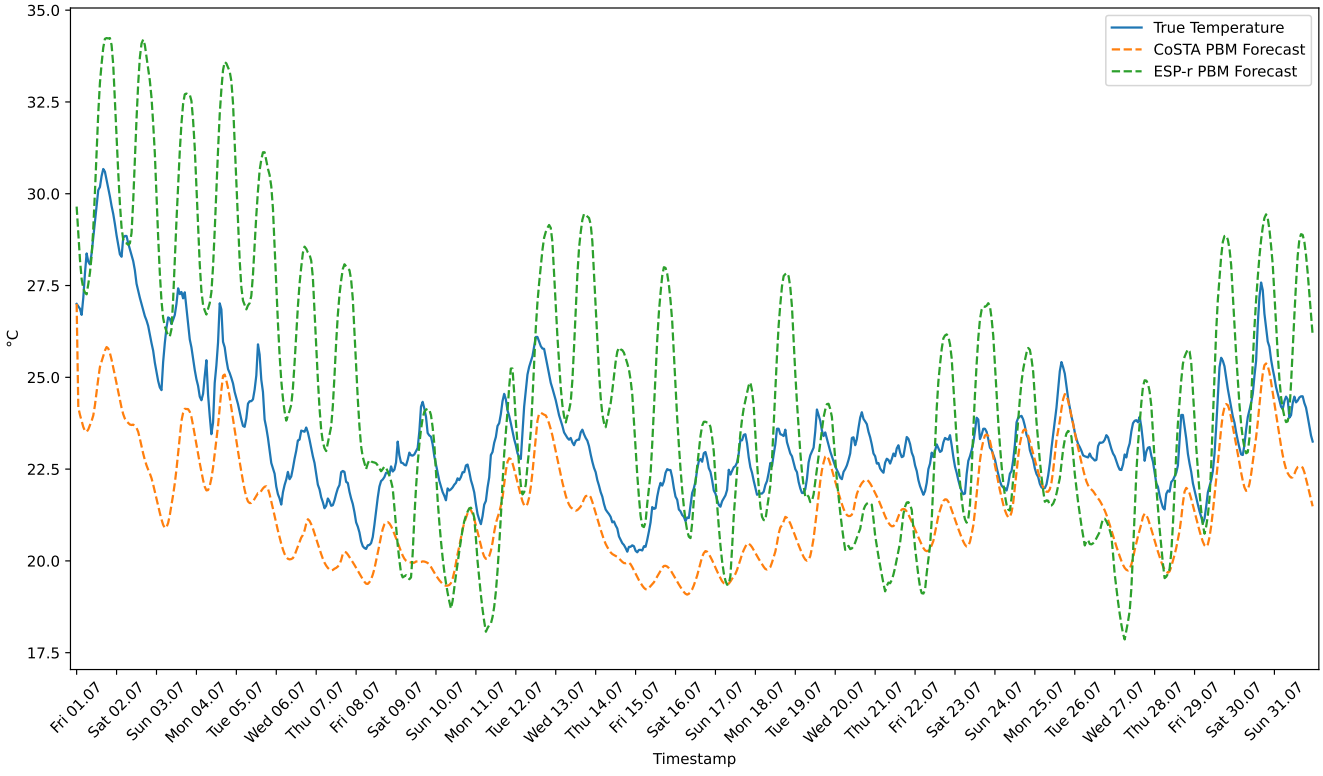


(d) 1st floor bedroom 1 forecasts

Figure 5.1.: PBM: True and forecasted temperatures of different rooms

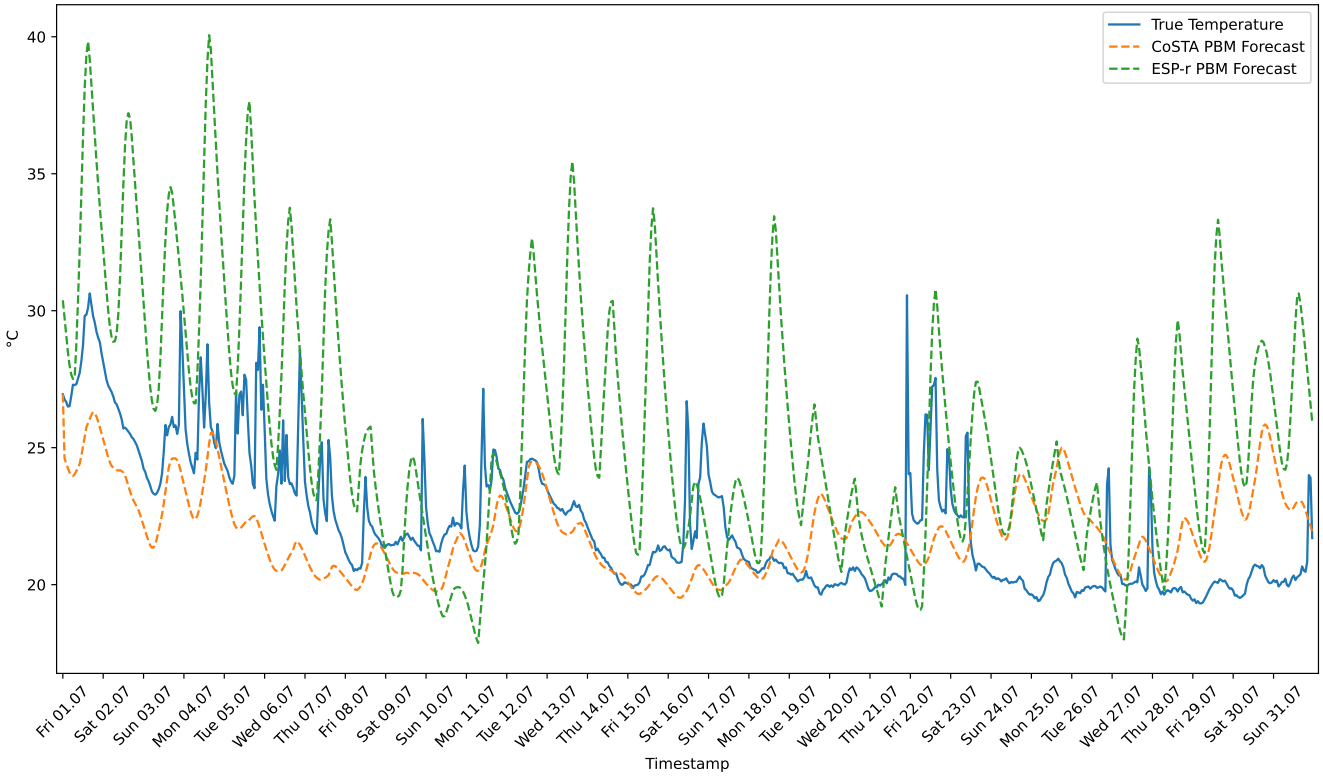


(e) 1st floor bedroom 2 forecasts



(f) 2nd floor living room forecasts

Figure 5.1.: PBM: True and forecasted temperatures of different rooms



(g) 2nd floor office forecasts

Figure 5.1.: PBM: True and forecasted temperatures of different rooms

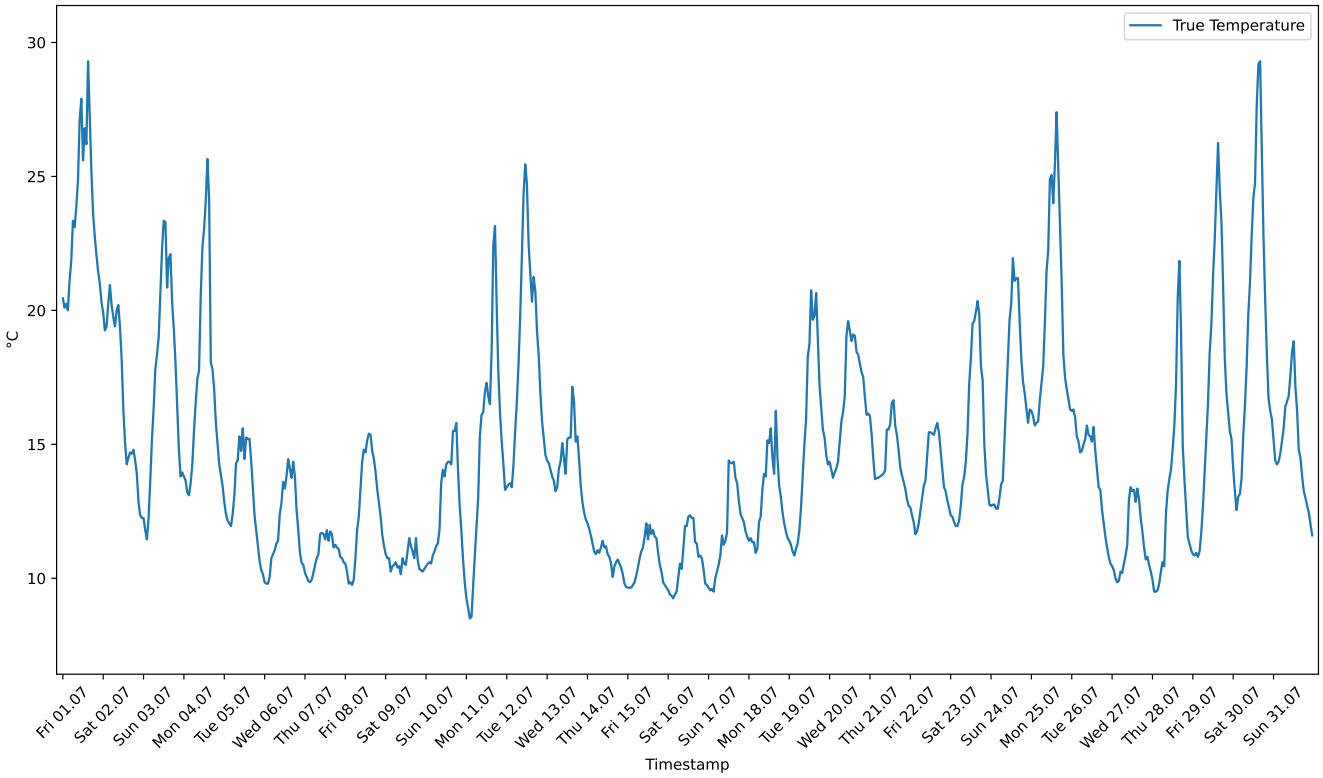


Figure 5.2.: Outdoor temperatures for July

5.1.2. Discussion

From the plots in fig. 5.1, it can be seen that both PBM models tend to follow the trends of the true temperature and are able to capture the general patterns of the temperatures of the asset. Both PBM models forecast the daily fluctuations in temperature, which also occur in the true temperatures. However, both models often fail to calculate the amplitude and offset of these fluctuations.

The most interesting model to discuss is the CoSTA PBM in order to assess its usability as the cornerstone of the CoSTA. The ESP-r PBM is primarily included as a benchmark, representing a more thoroughly developed PBM.

Interestingly, the CoSTA PBM outperforms the ESP-r PBM in terms of MAE, as evident in table 5.1. However, relying solely on MAE to assess model quality can be misleading in such a forecasting task. A pitfall of using MAE is that models that only forecast the general trends in the temperature may prevail. Optimizing electricity usage based on such forecasts is unlikely to achieve the desired outcome. It is therefore required to employ a comprehensive model able to model the inherent complexities of the problem to do electricity optimization. The complexity of the problem is immense. Nevertheless, the CoSTA PBM neglects a vast majority of this complexity.

Another interesting observation is that the very simplified CoSTA PBM forecasts essentially the same temperature evolution for every room, with a slight offset and time lag between the rooms. This general trend forecast is one factor that could lead to a lower MAE, while the forecast itself proves quite useless for electricity optimization.

On the more positive side, as indicated by fig. 5.1, the CoSTA PBM has demonstrated a certain level of efficacy in modeling the temperatures of the asset. It has also resembled, to some degree, the forecasts from the significantly more thorough ESP-r PBM, even though they often differ by some offset. Furthermore, it has successfully incorporated the asset's dominant temperature trends into its forecast, albeit simplified, due to the simplifications made during the modeling. For instance, the energy flux from the sun and internal heat gain were significantly simplified. However, despite these simplifications, the model still follows the principal trends of the temperatures of the asset.

The CoSTA PBM clearly models daily temperature trends, showing an increase during the day and a decrease at night. The timing of the peak and bottom temperatures also align well with the true temperature measurements. It also, to a certain degree, manages to model which periods the average indoor temperature will rise and fall, such as the somewhat steady decline in average indoor temperature for the first eight days of the month. This pattern correlates highly with the outdoor temperatures seen in fig. 5.2, which explains the model's successful forecasting of this period.

However, the average indoor temperature of a day does not always correlate with the outdoor temperature. For the last four days of the month, the outdoor temperature is among its highest, as seen in fig. 5.2. Despite the increase in outdoor temperature, the true temperature of, for example, the ground floor living room in fig. 5.1b stays relatively stable and does not increase. The exact reason for this behavior is tough to pinpoint due to the complexity of the problem. Likely explanations include some human interaction, such as nighttime ventilation or curtains to block the radiation out. None

of the implemented PBMs have knowledge about these interactions and are thus unable to forecast this. However, this increase in outdoor temperature is reflected in the true temperature of the 2nd floor living room in fig. 5.1f, where the true temperature rises. Consequently, the PBMs manages to model the temperature quite well.

From an optimization perspective, none of the models are especially accurate. Most of the time, they fail to forecast correct or close to correct values. A prime example is the 2nd floor office depicted in fig. 5.1g. The CoSTA PBM fails to forecast the temperature spikes occurring in this room. This is reasonable since these spikes do not correlate with the outdoor temperature and are seemingly not very correlated to the radiation. These are the only two time-dependent gains in the implemented CoSTA PBM model. Nevertheless, a forecasting model should be able to catch up on the evolution of temperature a bit better to optimize based on it.

However, some of the forecasts are relatively good, for example, the 2nd floor living room seen in fig. 5.1f. Here, the CoSTA PBM performs quite well, with more or less only a constant offset in the forecasted temperature for extended periods. Nevertheless, optimizing based on a constant offset is not very feasible, and some sort of rectifying component, for example, a data-driven component, rectifying this constant offset would be necessary for further optimization.

Based on these observations, it is reasonable to assume that the CoSTA PBM model itself is not good enough to be used as a cornerstone for further optimization of electricity usage. This was also assumed in advance due to all the simplifications made during the modeling. However, the objective of this model was for it to be used as the cornerstone of a CoSTA model, and from the plots in fig. 5.1 and the discussion above, it certainly seems like this model could be good enough to be incorporated into a CoSTA model.

Many of the previously discussed flaws, such as the inaccurate amplitude of daily fluctuations and constant offsets, are likely rectifiable by a neural network with relative ease. However, some of the other flaws, like the sudden spikes in temperature in the 2nd floor office and whatever caused the temperature in the ground floor living room to stay stable despite the increase in outdoor temperature, may be more challenging for a neural network to pick up on. One of the inherent advantages of data is its ability to provide a comprehensive overview of the problem at hand, while interpreting this data is often difficult due to the complexities and intricacies it often reveals.

To conclude, the CoSTA PBM seems to provide a sound basis for further incorporation into a CoSTA model. If successful, the CoSTA model can correct some of the previously discussed flaws and perform significantly better.

5.2. Data-Driven Modeling - DDM

5.2.1. Results

Figure 5.3 depicts how the error in the LSTM and the Transformer evolves with an increasing correction interval.

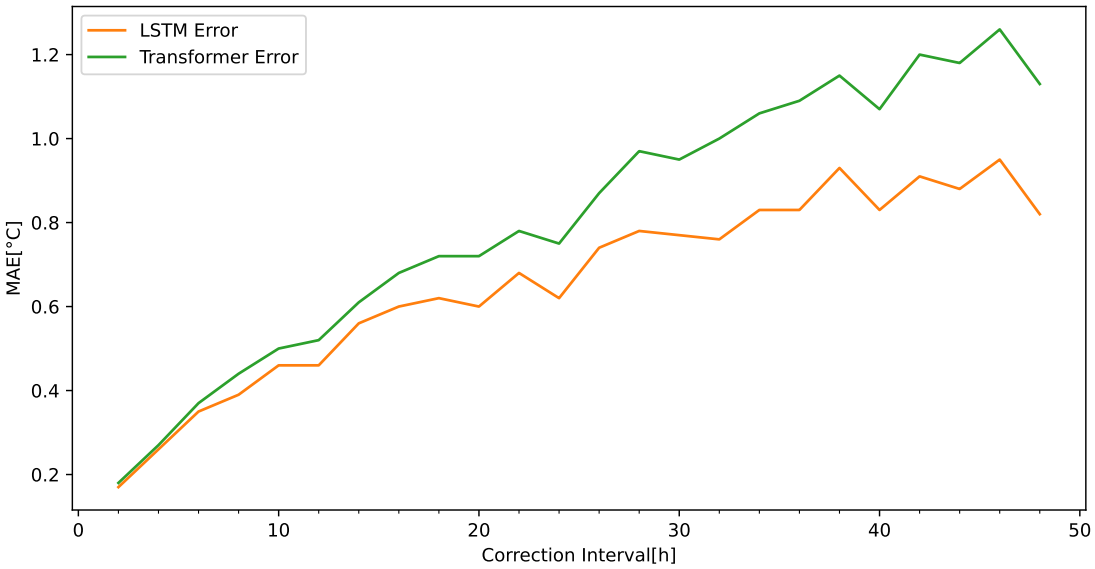
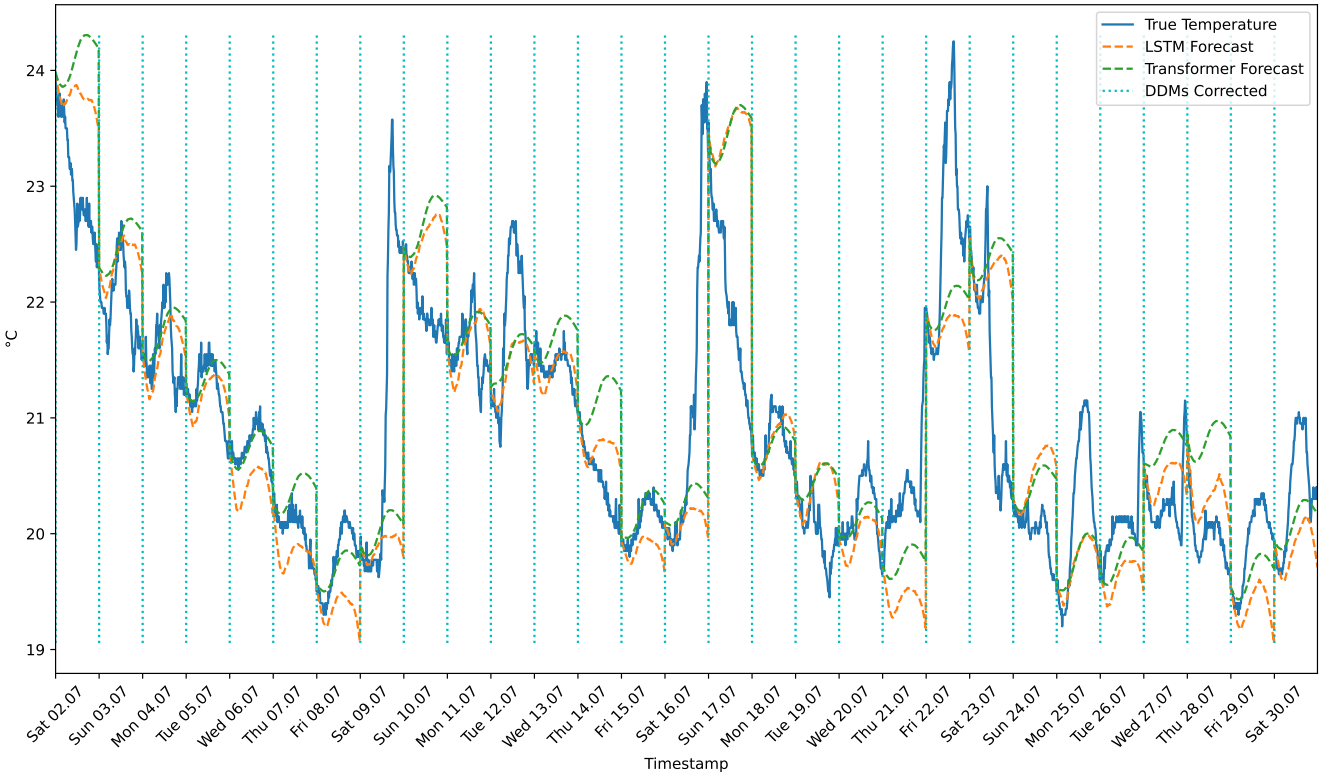


Figure 5.3.: DDM: MAE with increasing correction interval

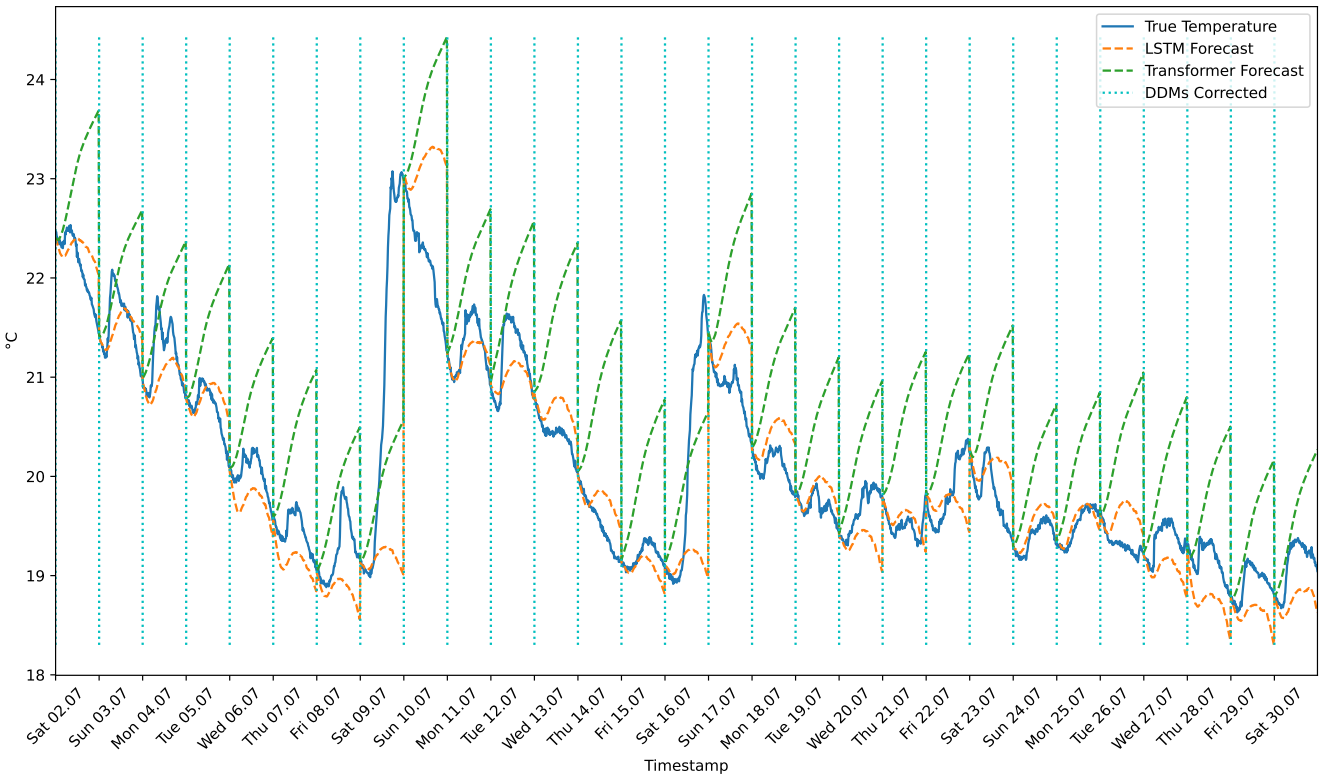
In order to examine the results more concretely, the correction interval was set to every 24 hours. Figure 5.4 displays how both of the DDMs forecast the temperatures for the given intervals of 24 hours, and table 5.2 describes how the error is distributed across the different rooms of the asset.

	Mean Absolute Error[°C]	
Room	**LSTM**	**Transformer**
Ground Floor Bedroom	0.51	0.52
Ground Floor Living Room	0.36	0.85
1st Floor Entrance Hall	0.45	0.46
1st Floor Bedroom 1	0.65	0.67
1st Floor Bedroom 2	0.58	0.60
2nd Floor Kitchen	0.85	1.05
2nd Floor Living Room	0.61	0.73
2nd Floor Office	0.98	1.03
Average MAE	**0.62**	**0.74**

Table 5.2.: DDM: Mean absolute error of the different rooms

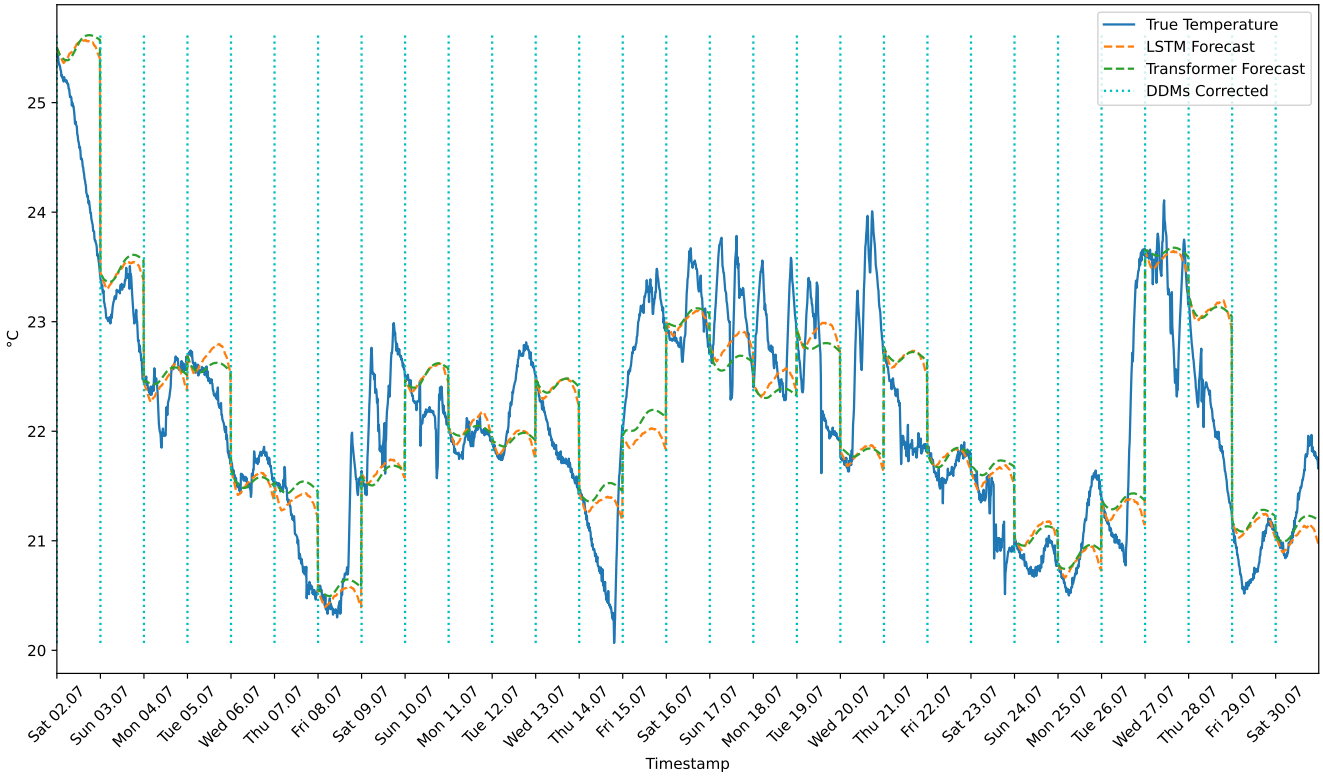


(a) Ground floor bedroom forecasts

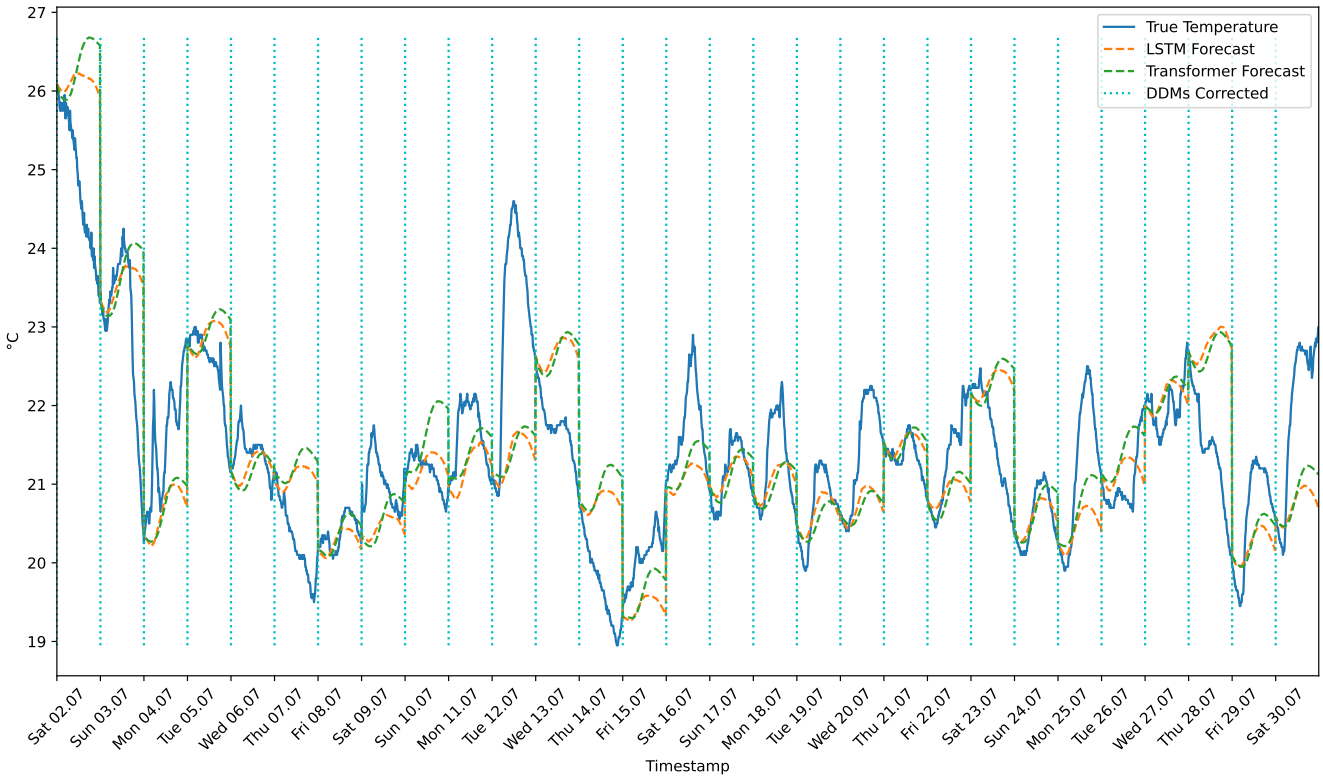


(b) Ground floor living room forecasts

Figure 5.4.: DDM: True and forecasted temperatures of different rooms

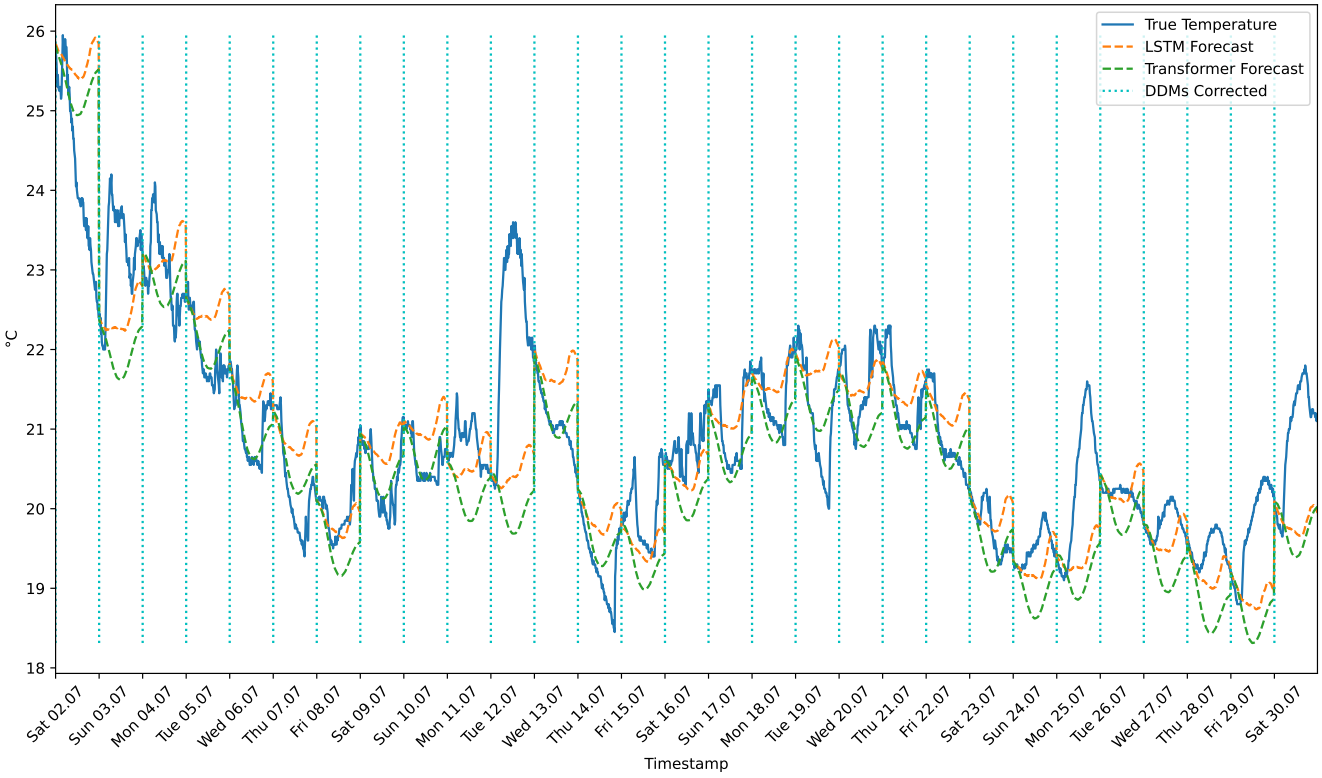


(c) 1st floor entrance hall forecasts

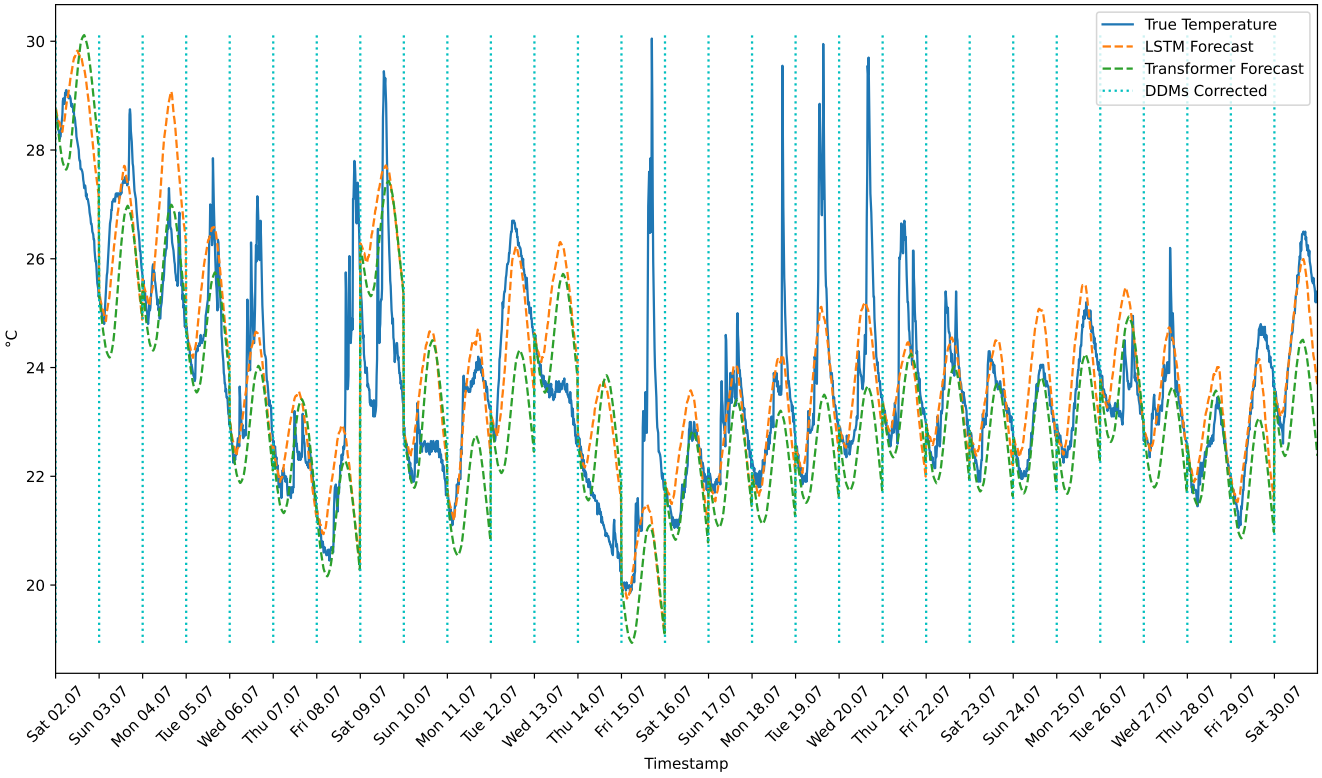


(d) 1st floor bedroom 1 forecasts

Figure 5.4.: DDM: True and forecasted temperatures of different rooms

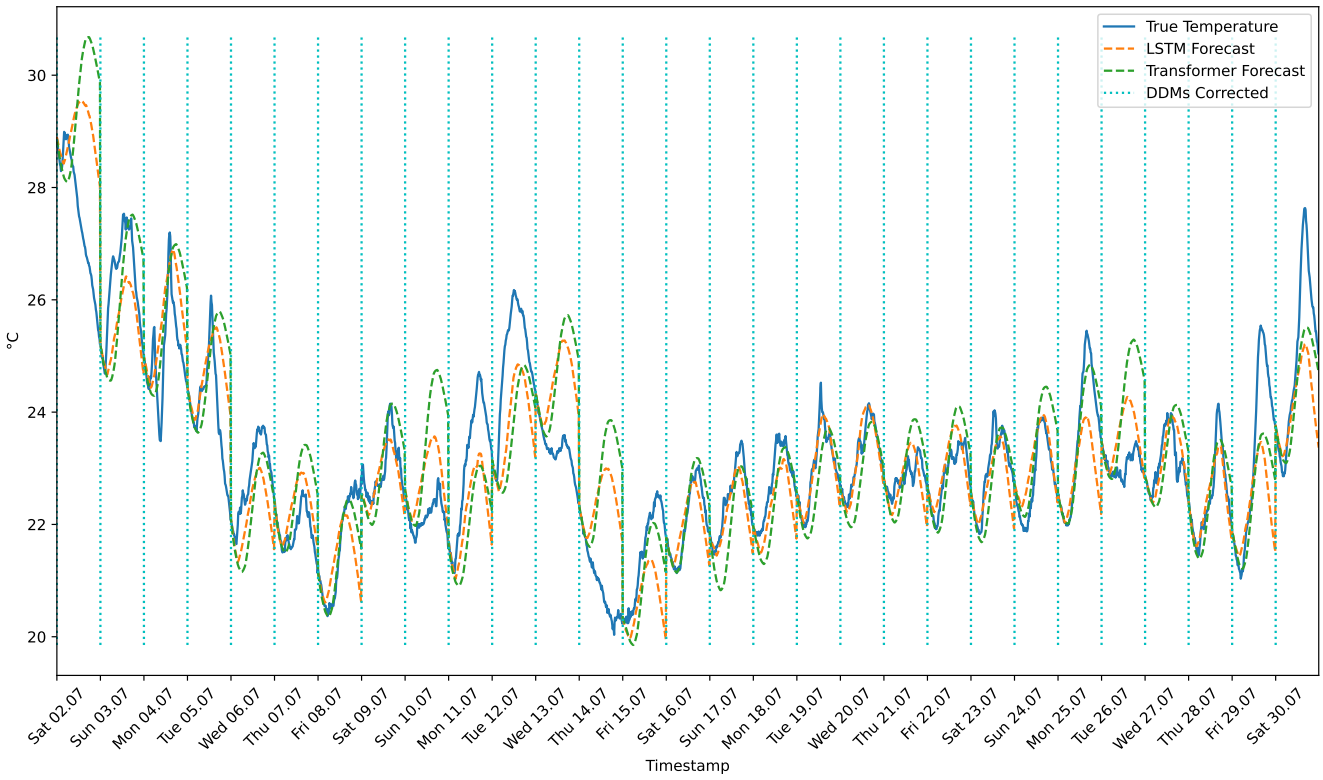


(e) 1st floor bedroom 2 forecasts

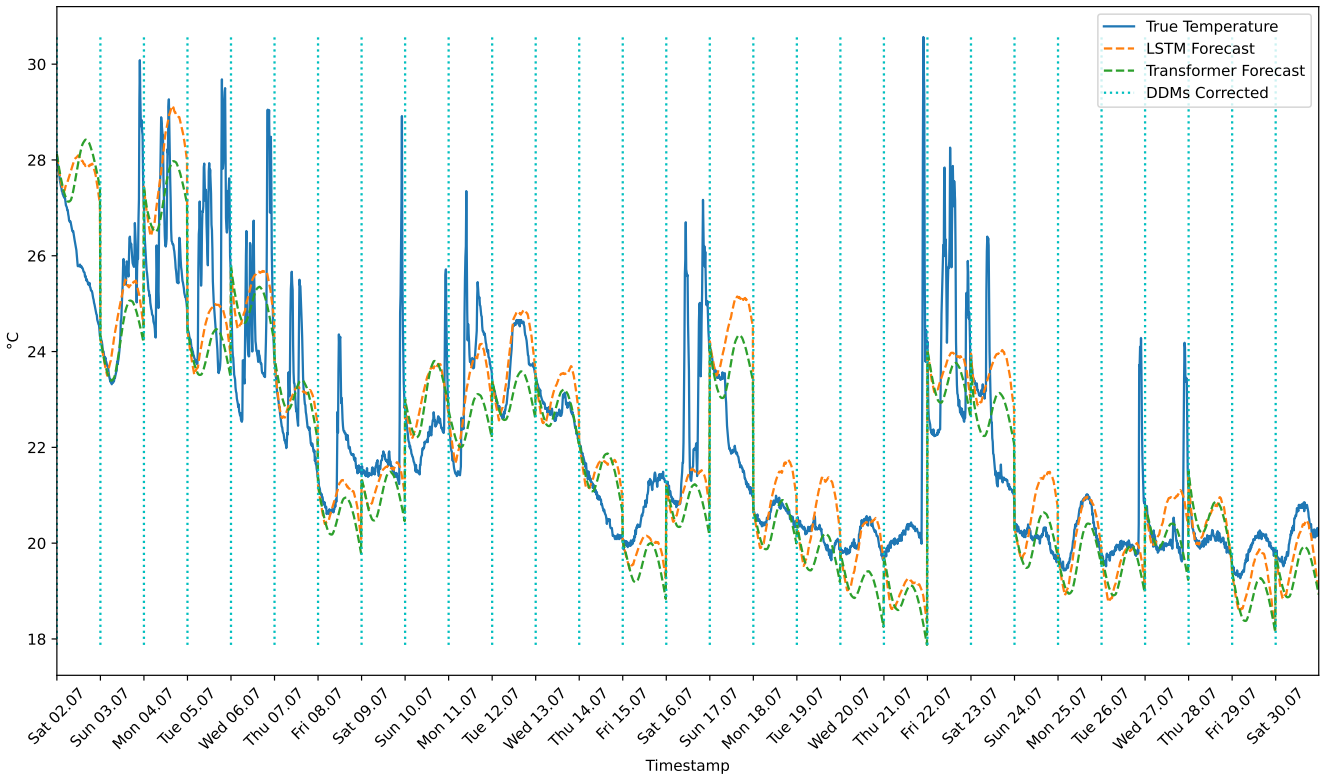


(f) 2nd floor kitchen forecasts

Figure 5.4.: DDM: True and forecasted temperatures of different rooms



(g) 2nd floor living room forecasts



(h) 2nd floor office forecasts

Figure 5.4.: DDM: True and forecasted temperatures of different rooms

5.2.2. Discussion

The error of both models grows more or less proportionally with the correction interval, as depicted in fig. 5.3. However, the LSTM handles the increasing correction interval better than the Transformer. One of the reasons for this might be the Transformer's forecast of the ground floor living room, as depicted in fig. 5.4b. The Transformer forecasts an increase in the temperature of this room no matter the circumstances. With an increasing correction interval, the deviation between the true and forecasted temperature increases, consequently influencing the MAE. The LSTM handles the forecasting of this room better.

There may be many reasons why the Transformer performs this poorly on the ground floor living room. One viable reason might be underfitting, which is one of the pitfalls when using early stopping, especially on a multiple-input multiple-output (MIMO) problem. As seen in the hyperparameter tuning of the Transformer in table 4.3, the early stopping found that the best hyperparameter for the number of epochs was three. The best number of epochs was decided based on the rolling forecast MAE on the validation set, which reasonably assesses the model's performance on unseen data. However, this MAE was calculated based on the average MAE across all the rooms the model was forecasting. It accepted underfitting or overfitting on some rooms as long as the average performance was good. This may have led to underfitting or overfitting for this particular room.

Nevertheless, this issue is hard to address for MIMO problems. One way to address this problem would be to implement separate models for each room, allowing the opportunity to tune the hyperparameters individually. However, this poses undesirable effects, such as the need for a significantly more comprehensive hyperparameter tuning, increased computational cost as there would be a need to train more models, and more data storage to store all the models.

Another way to address this issue would be to make a more sophisticated performance metric of the best validation MAE, for example, by imposing some restrictions on the accepted variability in performance across the different rooms. This would require a carefully constructed metric of the accepted variability, which might not be straightforward to derive due to the complexities of defining an acceptable range of variability.

From the plots in fig. 5.4, it is apparent that both models have learned a lot of the same dynamics. A prime example of this can be seen for the 1st floor entrance hall in fig. 5.4c. The forecasted temperatures from the models nearly perfectly resemble each other, even when they produce their worst forecasts compared to the actual measurements, for example, during their first 24 hours of forecasting. The forecasts from the LSTM and Transformer for the 1st floor entrance hall deviate by a mean of only 0.067°C for the entire forecasted month.

The fact that the two models are learning much of the same dynamics can mean two things; either the models are overfitting or underfitting on the same data, or the models are successfully approximating the underlying patterns and rules of the temperatures of the asset. However, given the significant difference in the architecture, it is more likely that the models are picking up on the actual underlying patterns and rules for several reasons.

Firstly, the models are forecasting using a rolling forecast and will, therefore, iteratively take their output as input. Such forecasting is notorious for its proneness to divergence and accumulating errors. The fact that none of the forecasts seems to diverge (except for the Transformer's forecast of the ground floor living room) substantiates this claim. Secondly, the models are extrapolating with regard to the season in the test set. Machine learning models are infamous for their underperforming extrapolation capabilities. Thirdly, the models operate with time-differentiated temperatures as inputs and outputs. This means that the output from the model must be integrated or added to the previous temperature to obtain the forecasted temperature. Such operations are notorious for their error-magnifying effect.

It is, therefore, rather unlikely that the two different architectures, with a significant difference in the underlying structure, are forecasting the same, relatively reasonable temperatures without actually resembling the underlying dynamics. Hence, this observation provides more certainty that the models are capturing the true dynamics of the system.

The fact that the models are learning so much of the same dynamics also indicates the robustness of the learning process. Since the models often arrive at the same conclusion, it is more likely that the dynamics they are learning are true features of the system and not an artifact of their respective model architecture.

Even though the models often forecast similar temperatures, the forecasts occasionally deviate quite significantly from the actual measured temperatures. An example of this can be seen in the greyed-out part of the ground floor bedroom forecast of the 17th of July in fig. 5.5. In this period, the models have an MAE of 1.46°C, and the error of the final forecasted value is 2.68°C for the LSTM and 2.79°C for the Transformer. The reason for this deviation is likely the rapid increase and decrease in the temperature, which started in the evening of Saturday the 16th of July. The models seem to struggle with forecasting such rapid and somewhat irregular changes in temperature, which is understandable. The models naturally have no precise knowledge regarding open windows or heat sources of the "future" they are trying to forecast. Therefore, irregularities like additional heating or cooling sources may offset the forecast.

However, the fact that some forecasts are significantly worse than others is to be expected, especially since the model has no explicit information about control inputs such as radiators, ventilation, and fireplace. Suppose the model had this information and was trying to forecast the temperature evolution based on a proposed future sequence of control inputs, which would be the case if the model was incorporated into an optimization algorithm. In that case, these errors would likely be reduced since the model would have explicit knowledge about the most significant driving forces of the temperature. However, forecasting errors would still have to be expected due to human interaction, such as additional people present imposing additional heat or opening windows or doors in an irregular pattern.

In the context of fig. 5.5, a plausible explanation might be that the ground floor bedroom had additional guests sleeping over from Saturday to Sunday, thereby imposing additional heat gain at the beginning of the night before opening a window so that

the temperature steadily started to decrease. Suppose there is no regularity or pattern concerning the time of such additional guests or interactions. In that case, it becomes impossible for the models to forecast these events without explicit information, and their forecast will be offset.

The fact that the models are struggling with spikes is apparent in several of the plots in fig. 5.4, particularly figs. 5.4f and 5.4h where the forecasts of respectively the 2nd floor kitchen and 2nd floor office are plotted. These rooms, which often experience irregular and rapid temperature changes due to events like cooking at different times of the day or working outside regular hours, are more challenging to forecast than rooms with more apparent recurring daily trends. This difficulty can also be observed in table 5.2, where these two rooms have the highest MAE.

Another factor that contributes to this irregularity is the choice of forecasting month. July is the most irregular month throughout the year, with a significant portion of the population going on vacation, thereby changing their regular day-to-day schedules. The models are trained on data from all months other than July - which typically have more recurring daily schedules - and will naturally struggle to accurately forecast the more irregular patterns seen in July. An additional year of training data would likely benefit the models, as they would no longer need to extrapolate for July and might be able to model some of the irregularities July introduces better.

These observations show that the models work best during stable periods with clearly recurring daily trends. Figure 5.6 depicts such a period with the forecasted temperatures of the nine days between the 16th of July to the 25th of July of the 2nd floor living room. This is one of the best periods of forecasts and powerfully illustrates the potential of these models. The LSTM model achieves an MAE of only 0.27°C during the entire period, while the Transformer achieves an MAE of 0.39°C. This period has a clear daily trend; the most significant difference between the days is the starting temperature and magnitude of the daily oscillation.

To conclude the DDM discussion, there is no doubt that the models have learned some of the dynamics of the temperatures of the asset, even though they are struggling with any irregularity or spike. From the observations and discussions above, the LSTM model performs better overall than the Transformer model for this forecasting task, which is also the case when looking at table 5.2. The LSTM model also appears relatively stable, with 80% of its forecasted values having an absolute error of less than 1°C. Therefore, it is very likely that such a model would have the potential to work well as a cornerstone for optimizing electricity cost or usage. This topic is further delved into in section 5.4.

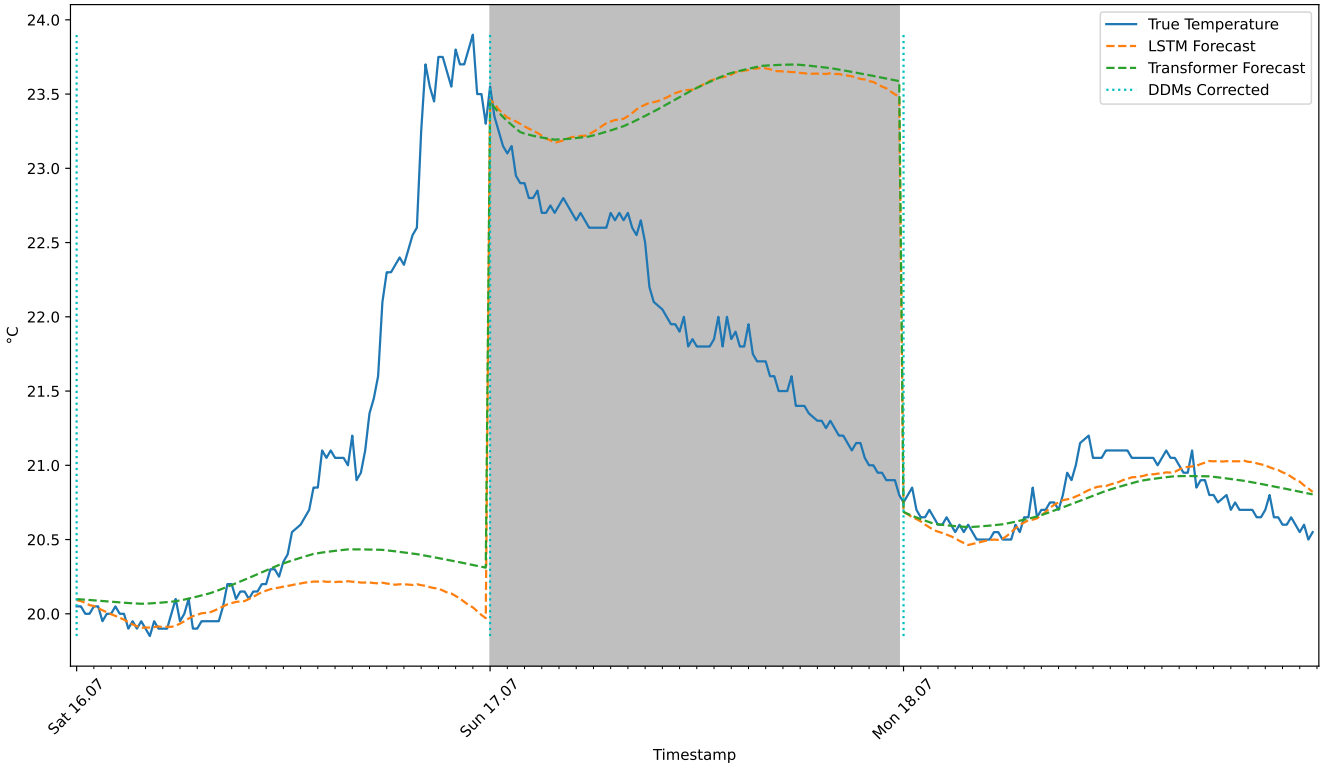


Figure 5.5.: DDM: Ground floor bedroom forecast of the 17th of July

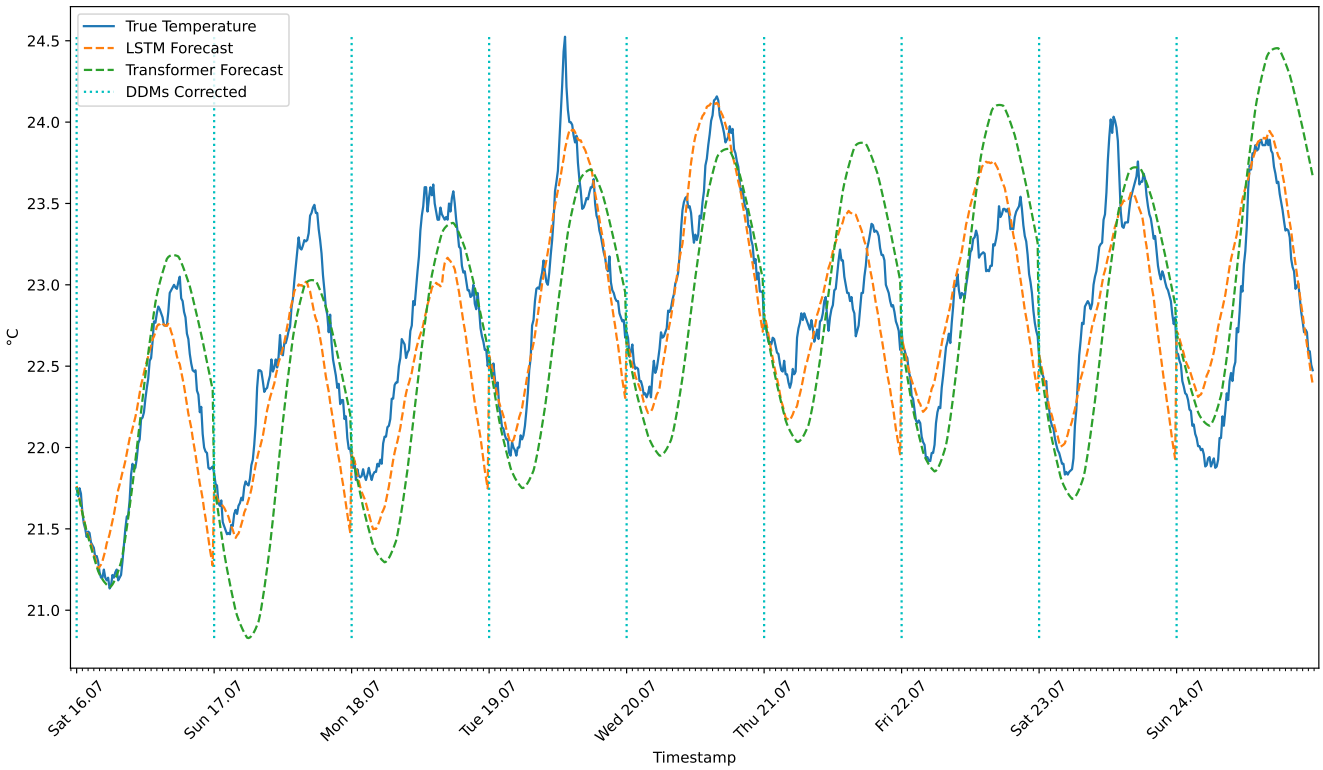


Figure 5.6.: DDM: 2nd floor living room forecast of the period 16th to 25th of July

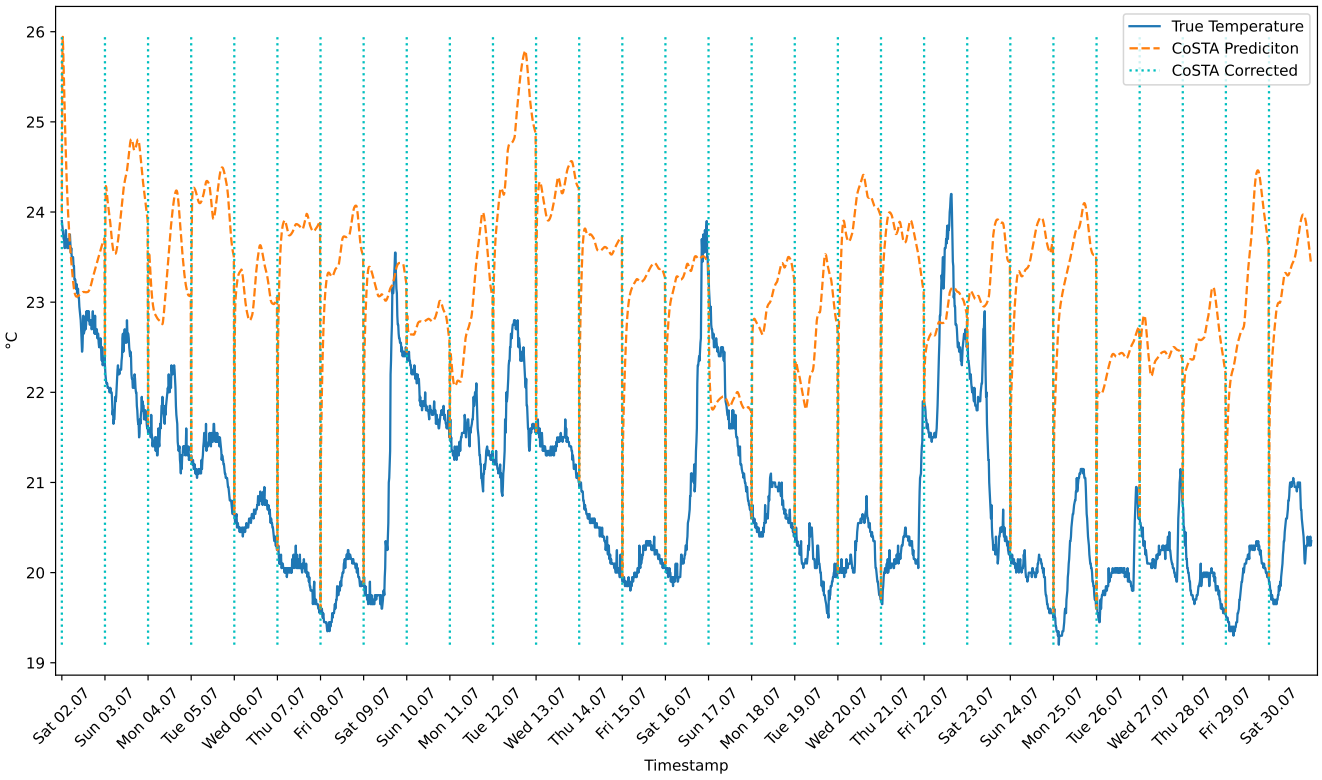
5.3. Hybrid Analysis and Modeling - HAM

5.3.1. Results

Table 5.3 describes the forecast MAE of the implemented CoSTA model. The corresponding forecasts are plotted in fig. 5.7, together with the true temperature of the same rooms. The 2nd floor kitchen is neither included in this model for the same reasons described for the PBMs in section 5.1.

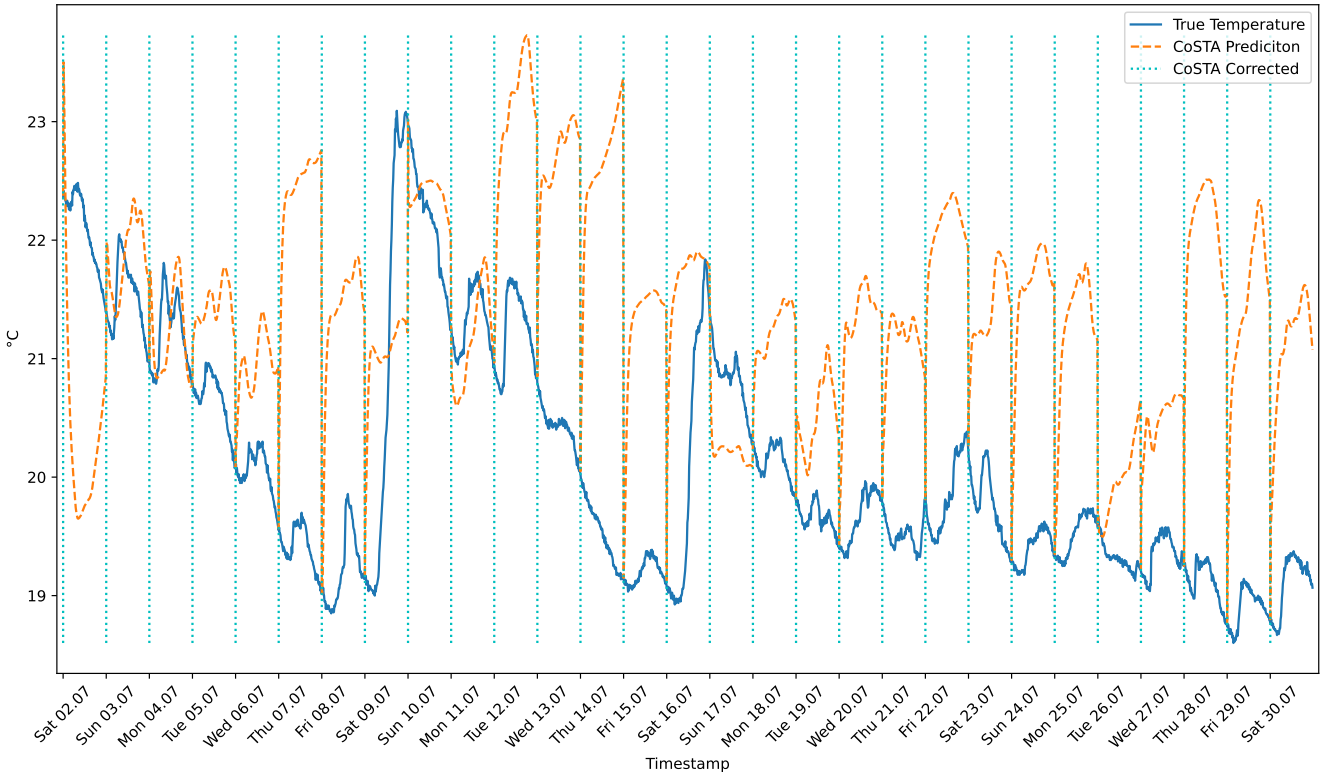
	Mean Absolute Error[°C]
Ground Floor Bedroom	2.46
Ground Floor Living Room	1.56
1st Floor Entrance Hall	1.27
1st Floor Bedroom 1	0.81
1st Floor Bedroom 2	0.88
2nd Floor Living Room	1.09
2nd Floor Office	2.03
Average MAE	**2.06**

Table 5.3.: HAM: Mean absolute error of the different rooms

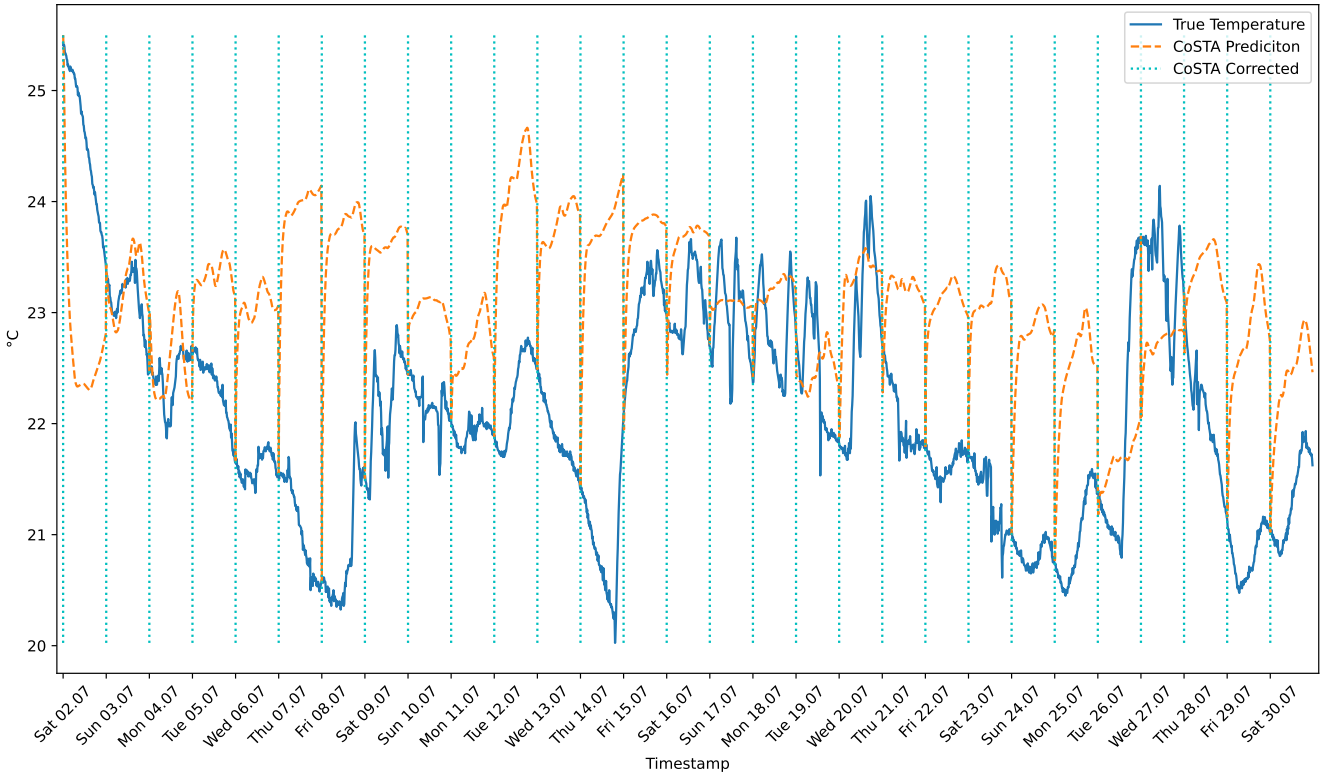


(a) Ground floor bedroom forecasts

Figure 5.7.: HAM: True and forecasted temperatures of different rooms

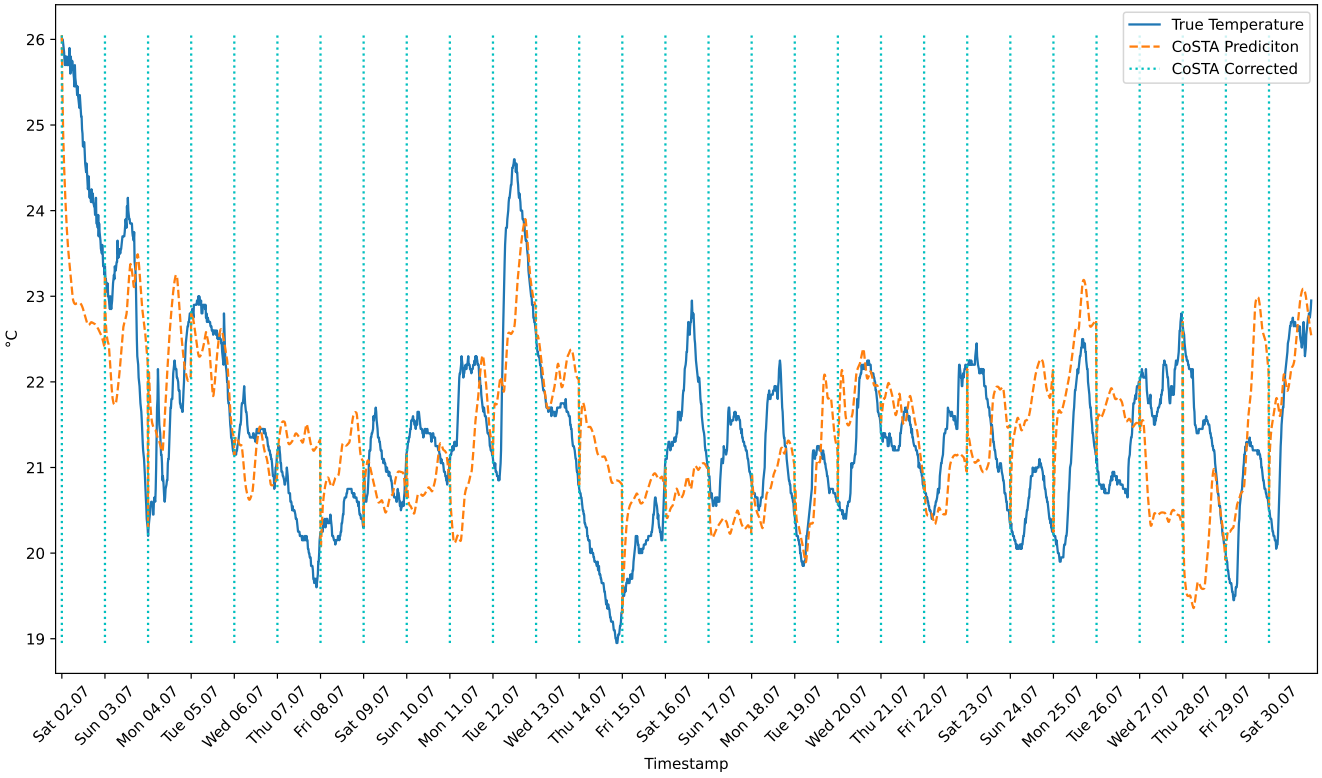


(b) Ground floor living room forecasts

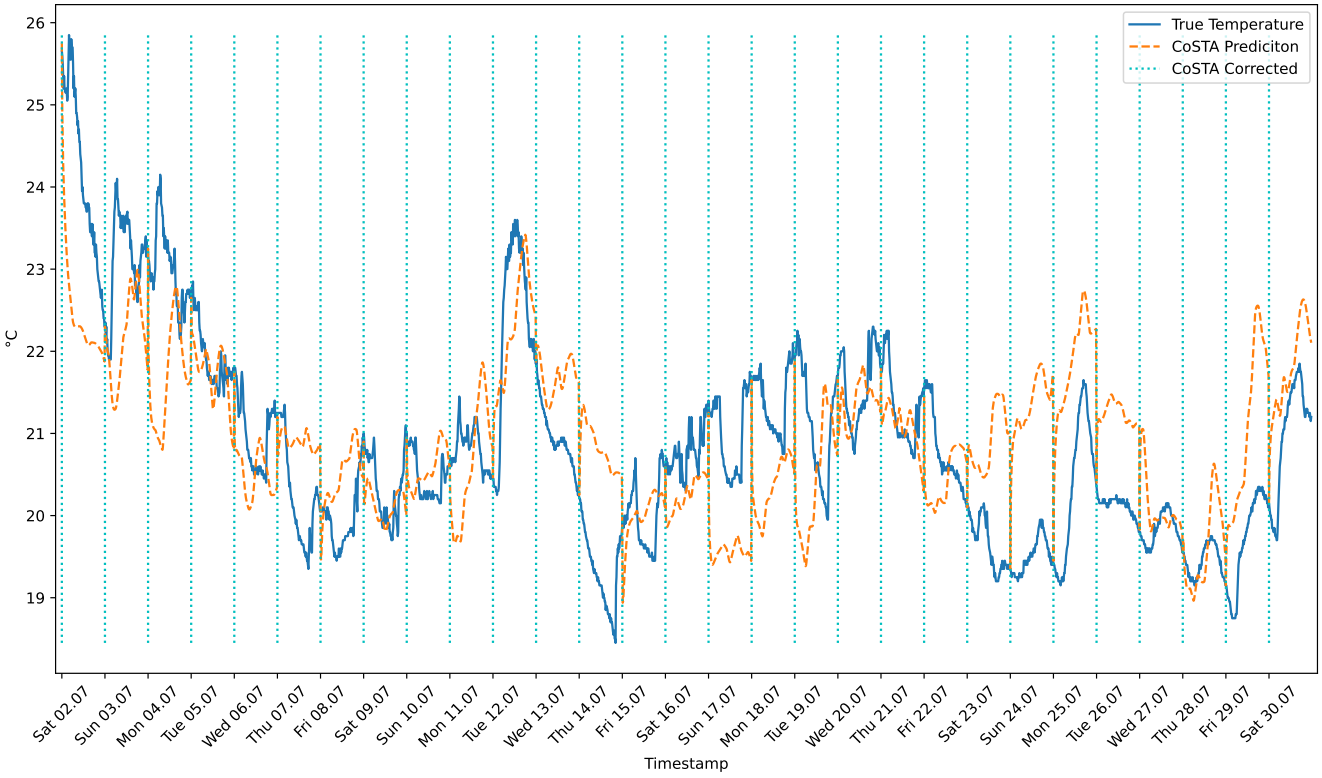


(c) 1st floor entrance hall forecasts

Figure 5.7.: HAM: True and forecasted temperatures of different rooms

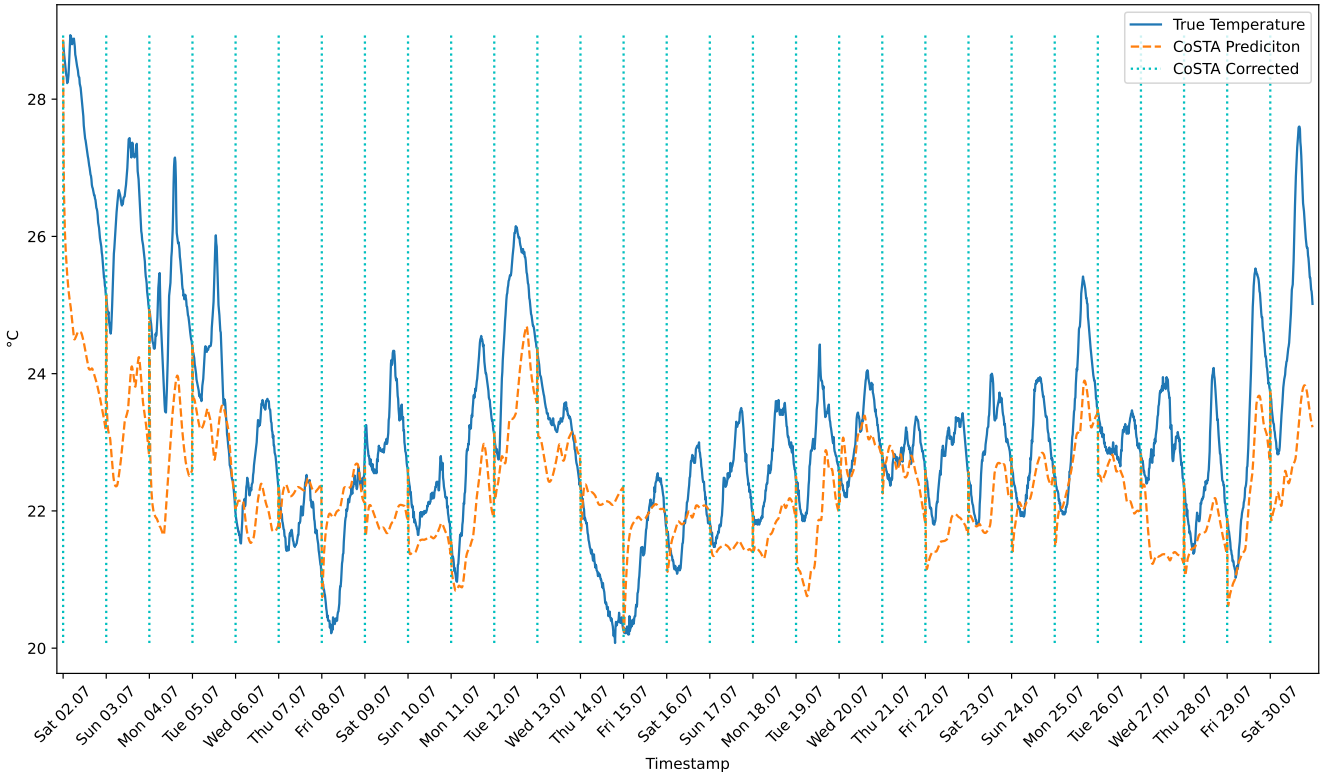


(d) 1st floor bedroom 1 forecasts

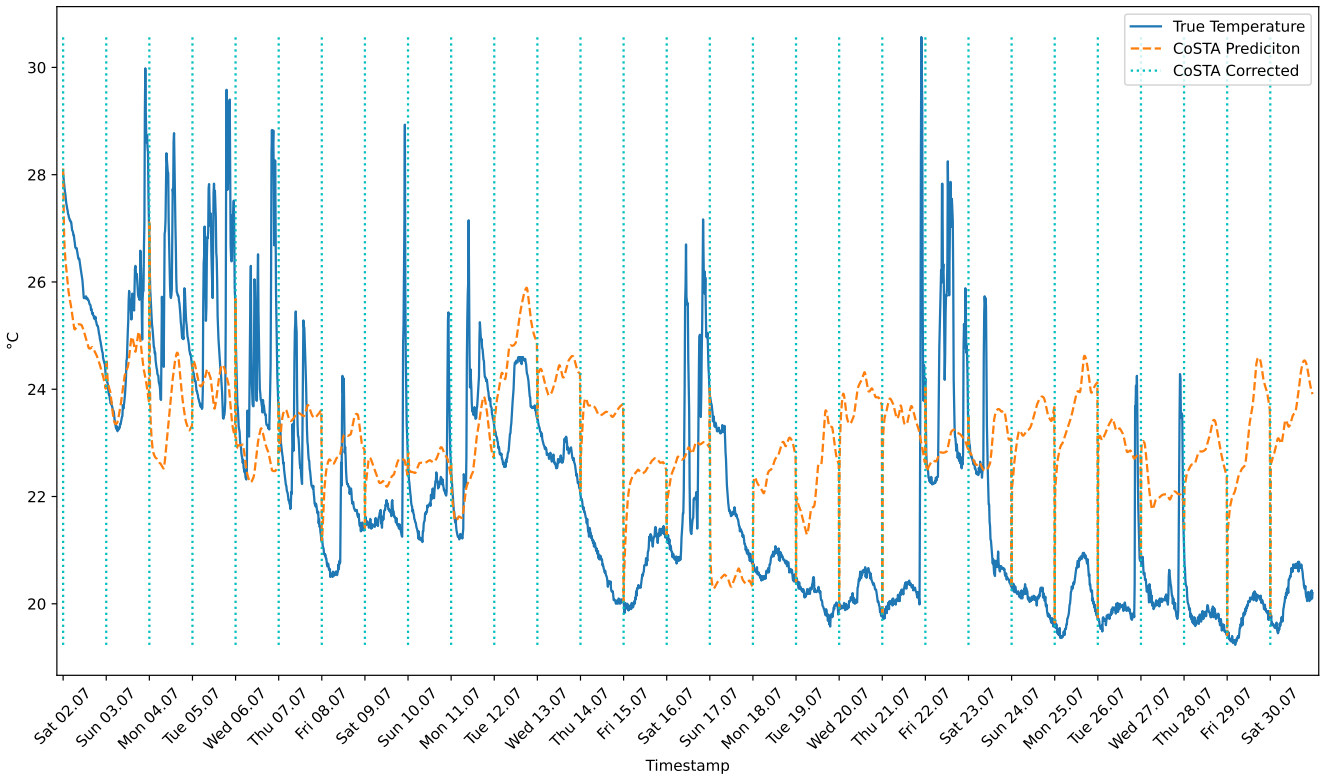


(e) 1st floor bedroom 2 forecasts

Figure 5.7.: HAM: True and forecasted temperatures of different rooms



(f) 2nd floor living room forecasts



(g) 2nd floor office forecasts

Figure 5.7.: HAM: True and forecasted temperatures of different rooms

5.3.2. Discussion

As fig. 5.7 reveals, the implemented CoSTA model's performance falls short of expectations on this forecasting task, even after an extensive training period and hyperparameter tuning to make it perform as well as possible. Even though some of the forecasts resemble the dynamics going on in the asset, as can be seen in the 1st floor bedroom 1 forecast in fig. 5.7d, the model does not manage to perform particularly well, especially not from an optimization perspective.

As seen in figs. 5.7a and 5.7b, the forecasts for the rooms on the ground floor are highly inaccurate. The model more or less always forecasts a temperature of several degrees too high, even after being corrected with the true temperature. Hence it is by no means feasible to impose any optimization based on these forecasts.

There could be several reasons for these suboptimal forecasts. The apparent reason is that the training routine has decided on weights and biases to minimize the loss over the entire training set. However, these weights and biases do not adequately represent the general error in the underlying equations. The underlying reason(s) why this occurs is not readily apparent, which is one of the main drawbacks of "black box" methods such as neural networks, whose inner workings are not easily understandable. Furthermore, the complexity of the problem is enormous, and the error in the underlying equations is thus also inherently difficult to predict.

This suboptimal performance of the DDM part may have several explanations, including incorrect weight-and-bias initialization, inadequate learning rate, improper regularization, suboptimal dataset, improper feature normalization, or inappropriate model architecture.

The network's weights and biases were initialized using the recognized He initialization, the same scheme utilized by the DDM LSTM. It is thus unlikely that this is the leading cause. Various learning rates were also experimented with, but none provided better results. The network is also heavily regularized, as seen in table 4.7. This regularization may naturally lead to underfitting, but the regularization parameters were also experimented with without enhanced performance.

Regarding the dataset and normalization of features, the LSTM input to the CoSTA was identical to the LSTM input in the DDM LSTM. The only difference between the inputs to the DDM part of the CoSTA and the inputs to the DDM LSTM is that the FFNN layers of the CoSTA received an additional 26 $\vec{T}_{room}$ and $\vec{T}_{wall}$ features. These features were also normalized, and as long as they appear in the same range as the forecasts produced on the training set, they should be z-score normalized. Hence, this is neither believed to be the leading cause of its inability to learn.

The architecture of the CoSTA unit has also proven to work relatively well on this forecasting problem, as seen in section 5.2. Despite the predictive task of the DDM in a CoSTA model differing significantly from the predictive task of a pure DDM, it is believed that both DDMs depend on a lot of the same dynamics and require models of a relatively similar capacity. The capacity of the CoSTA DDM was also explored, but it did not significantly impact the performance. Therefore, the choice of DDM model architecture is neither believed to be the leading cause of the suboptimal performance.

The fact that the DDM part of the CoSTA is not the final output layer may be one of the reasons for this suboptimal performance. This design choice, as advocated by Blakseth et al. in their original CoSTA model, was made to enhance interpretability[13]. However, this introduces additional complexity to the prediction task of the DDM. For a standalone DDM, it is easier to forecast a temperature closer to the most recently observed temperature than for a CoSTA DDM, which has to predict a corrective source term enabling the PBM to forecast a temperature within the same range. This trade-off between interpretability and the DDM's prediction complexity enhances the CoSTA's interpretability. Nevertheless, this design choice may lead to a more challenging learning process for the DDM.

The choice of letting the error gradients handle the calculation of correct labels for the corrective source term based on the output temperature from the PBM, as described in section 2.4, may also have further complicated the training process. The longer the error gradient has to traverse backward on the computational graph to update the weights, the more likely it is that the gradient will vanish, meaning that it will not be able to update the weights appropriately. This may be one of the causes for the inability to accurately learn the corrective source term.

Another potential reason for the model's inability to accurately learn the corrective source term may be the number of labels compared to the number of targets the DDM part is trying to predict. As seen in fig. 3.1, the asset has measurements of temperatures in seven of the 13 rooms. These measurements are sampled every 15 minutes. The CoSTA model is, however, trying to forecast the temperature in all 13 rooms of the asset. Since the CoSTA model utilized a step size of 60, the model has to be solved iteratively 15 times before reaching the next sampled measurement. Additionally, the DDM part of the CoSTA is trying to predict the source term of $\vec{T}_{wall}$ as well as $\vec{T}_{room}$. This results in the DDM attempting to predict 26 targets 15 times but only having seven labels to evaluate its predictions. This disparity makes it more challenging to find optimal weights. It should be possible, given that the dynamics of the 26 targets likely will be similar, and the seven labels may be able to describe these dynamics, but it is not optimal. An idea could be to predict only the source term of $\vec{T}_{room}$, thereby halving the number of targets, but it is highly uncertain whether this will lead to any improvements.

It is also reasonable to believe that the error in the underlying equations of the PBM is somewhat seasonally dependent, making the error non-stationary. As previously mentioned, neural networks are notorious for their underperformance on non-stationary data. Some of this non-stationarity was in the pure DDMs counteracted by time-differentiating the features and labels. However, no measures have been taken to counteract this non-stationarity in the CoSTA model except time-differentiating the input to the LSTM, as the model's nature makes the non-stationarity more difficult to counter. This could be a factor in why the DDM part of the CoSTA is not able to disclose the source term of the underlying equations.

Another weakness of the PBM model implemented is its reliance on the $\vec{T}_{wall}$ variables. The true values of these variables are never known, and when "correcting" the forecasts from the model every 24 hours, these values are approximated by doing a "warm-up sequence" on the previously measured temperatures. This sequence will approximate the

variable's value, but it is not highly accurate, and the model could benefit from having fewer time-dependent unknown variables. However, the dynamics of these temperatures are an utterly complex problem, and it is unlikely that a simpler PBM could adequately model the temperatures of the asset.

Therefore, another potential reason for the inability to learn is that the PBM part is too superficial in describing the temperatures of the asset for the DDM part to determine its corrective source term. The DDM part certainly seems incapable of disclosing the correct corrective source term of the underlying equations from the PBM. This suggests that the PBM, although seemingly representing the temperatures of the asset effectively in fig. 5.1, might not be an acceptable representation of the underlying patterns and rules of the system for the CoSTA. This would mean that the inherent error in the underlying equations is not generalizable enough for accurate prediction based on the given input features. This could also mean that the error in PBM's underlying equations varies significantly across seasons due to non-stationarity, making it harder to disclose the corrective source term.

This observation suggests the need for a more thoroughly developed PBM to improve the forecasts, factoring in more of the influencing elements of the temperatures of the asset. Undoubtedly, the current PBM relies on numerous assumptions, especially regarding the heat gain and the energy flux from the sun, which might make the error in the underlying equations harder to generalize. However, a more thoroughly developed PBM, able to factor in more dynamics, could make the error in the underlying equations easier to disclose and possibly diminish the seasonal dependencies in the source term. Consequently, this could enhance the model's overall performance.

From this discussion, it is clear that pinpointing the exact reason for this suboptimal performance is challenging. Therefore, significant work would have to be done regarding the previously discussed points to improve the model. A starting point would be to investigate the PBM more thoroughly and alter the model based on the findings.

To conclude, the core concept behind the CoSTA architecture holds significant promise for the future of forecasting models. This concept enhances the interpretability of the model and may improve the model's overall performance, thereby representing a significant idea for advancing forecasting methodologies. However, despite these enhancements and the fact that the CoSTA architecture has a solid theoretical foothold[12], it demands an adequate PBM to make the error in the underlying equations generalizable, thereby disclosable for the DDM. This seems not to be the case for the PBM proposed in section 4.3.

5.4. Applicability for Electricity Optimization

Given the CoSTA model's failure to deliver good forecasts, as seen in fig. 5.7, and the PBMs' inability to produce accurate forecasts applicable for electricity optimization purposes, as evidenced in section 5.1, it appears the DDM paradigm has emerged as the most effective paradigm for this forecasting task at present.

As outlined in section 5.2, the LSTM slightly outperforms the Transformer regarding reliability and accuracy. Consequently, the LSTM model, of all the implemented models, offers the best forecast and is the model which should be incorporated into an electricity optimization algorithm to assess potential savings in electricity cost or consumption. However, to fully perform this function, the model would require adaptation to accommodate control inputs, like the impacts from radiators or ventilation systems.

Incorporating control inputs into the model would likely enhance its overall performance. The purpose of such a model in the electricity optimization context would be to examine and decide upon different future control sequences, meaning that the model would have explicit information about the future of the most significant driving forces of the asset's temperatures, namely the control inputs. This information should allow the model to predict rapid increases and decreases in temperature more precisely, such as the sequence observed in fig. 5.5. However, as previously discussed, forecasting errors should still be expected due to human interaction.

Undoubtedly, strategically timing electricity usage can result in a lower electricity bill due to the inherent fluctuations in the electricity price. The electricity price in Norway is always known for the next 24 hours, which makes it possible to optimize electricity usage based on the known electricity price. An example of the potential savings due to fluctuations in the electricity price can be given for eastern Norway on the 23rd of May, 2023. The electricity spot price evolution for this day is depicted in appendix A.2. Between 8 AM and 9 AM, the price was 1.10 NOK/kWh, while between 3 PM and 4 PM, the price was 0.06 NOK/kWh. Hence, electricity is 95% cheaper at the lowest price compared to the highest[9]. With the increasing penetration of wind energy and its inherent volatile nature, the energy price may fluctuate even more on a daily basis[10, 11]. Therefore, it is reasonable to assume that the potential savings might be significant, given a good forecasting model and a proper optimization algorithm.

An example of such an optimization sequence that would occur every weekday for a residential house is the period between the inhabitants leaving the house for work and when they return. Significantly simplified, the temperature in the house's different rooms does not matter in this period as long as the house is in a desired thermal state when the inhabitants return. For example, say the inhabitants always leave for work at 7 AM and return home at 5 PM. This absence gives 10 hours of optimization space, where the electricity consumption can be optimized in any way, provided the house returns to the desired thermal state by 5 PM.

This approach also has vast potential from the climatic perspective. Optimization based on total electricity usage is no more challenging than adjusting consumption in response to price fluctuations. An example of such a situation is when employees arrive at work. A simple thermostat may cause the temperature to overshoot before the ventilation kicks

in. Hence electricity is wasted on both heating and cooling. However, a sophisticated model may forecast this, and the control system may be adjusted accordingly. Such a model can also optimize electricity consumption overnight when no one occupies the building, consequently decreasing energy usage.

With an appropriate optimization algorithm, this approach even has the potential to combine the two optimization strategies, optimizing based on both total electricity usage and electricity cost. This approach would require a weighting of the importance of the two factors. By assigning a weight to each factor according to their priorities, electricity consumers can tailor the optimization to their specific needs, whether primarily concerned with reducing electricity costs, minimizing consumption for environmental reasons, or achieving a balance between the two.

Undoubtedly, some of the forecasts produced by the LSTM are good enough for further electricity optimization. A great example is the nine days depicted in fig. 5.6. For this entire period, the LSTM is essentially spot on. Hence it should be possible to reduce energy costs or consumption by optimizing based on this.

However, as seen in section 5.2, the LSTM does not always forecast adequate temperatures for optimization purposes. In addition, the interpretability of the model is low due to its "black box" nature, which undermines its trustworthiness. Its interpretability may pose a problem when integrating it into a real-world application, especially if it is a high-stakes application, as there is no guarantee that the model's forecast is reasonable. It might therefore be needed to implement some confidence measures on the forecasts to make the optimization algorithm able to assess the reliability of the forecast. History has shown that one of the most essential factors for humans to change their way of doing things is that it is not at the expense of their comfort in addition to some reward. Therefore, maintaining satisfactory thermal comfort during these optimizations is crucial.

5.5. Lessons Learned

This work has highlighted several important lessons learned about modeling physical processes. In hindsight, the following list presents some of the most important lessons learned and details that would have been done differently if this study was to be repeated.

1. The complexity of the indoor temperature forecasting problem is immense. Due to the numerous influencing factors, a pure PBM will not be able to forecast the temperatures of a building accurately from an optimization perspective. The fact that data may manifest all the influencing factors of the temperatures thus seems like an important prerequisite for accurate forecasts.

2. The DDMs prosper on the premise that data may manifest all the influencing factors of the temperatures. However, the notorious fact that DDMs are often as interpretable as a "black box" hurts their trustworthiness.

3. The HAM paradigm contains powerful tools for enhancing the interpretability and trustworthiness of accurate forecasting models. However, as this work has shown, they require significantly more domain knowledge and some level of synergy between the PBM and the DDM part, which is not always trivial to foresee and derive.

4. If the DDMs were to be implemented again, it would be interesting to experiment with the water and humidity sensors in the dataset. As seen in appendix A.1.3, the asset also has measurements of these factors, which were not included in this study. These sensors may reveal patterns regarding cooking, showering, or people present in the asset, which may influence the evolution of temperature, such as the spikes in temperature experienced in the 2nd floor kitchen. It would also be interesting to include the radiation data from NCCS in the dataset of the DDMs, as it was utilized in the CoSTA.

5. If the CoSTA was to be implemented again, the PBM model would have been developed in a more sophisticated manner. A more sophisticated representation of the heating load and the energy flux from the sun would benefit the CoSTA. Additionally, the asset has installed balanced ventilation, whose datasheet is attached in appendix A.1.4, which could be beneficial to include in the PBM, making it easier to disclose the underlying source term.

6. Tracking and measuring of control inputs at the asset would likely also benefit the DDMs and the CoSTA, as these features are some of the most important driving forces in the evolution of indoor temperature. This would also allow for assessing the potential savings attainable from such models by incorporating them into an optimization algorithm. However, the asset is not measuring these variables at present.

6. Conclusion and Further Work

This chapter will first conclude on the research questions proposed in section 1.2.3 before discussing potential avenues for further work.

6.1. Conclusion

- **Which PBM, DDM, or HAM architecture forecasts the evolution of the indoor temperatures of a building most accurately?**
  Based on the results presented in chapter 5, it is apparent that the DDM paradigm managed to forecast the evolution of the indoor temperatures of a building most accurately. Within the DDM paradigm, the LSTM provided the most accurate and reliable forecasts and is thus recognized as the most accurate architecture for this forecasting problem. However, the HAM model implemented did not function as desired and may not accurately represent the full potential of this architecture or paradigm. Therefore, the HAM paradigm cannot be completely dismissed.

- **How reliable are the forecasts from the most accurate model?**
  From the plots of the LSTM forecasts in fig. 5.4, the forecasts from this model seem pretty reliable. The forecasts often align closely with the true temperature and rarely deviate far from the true temperature. It is also reasonable that the forecasts occasionally deviate from the true temperature, as the model has no explicit knowledge about future control sequences or human interaction, which might significantly impact the complex problem of temperature evolution. Nevertheless, the model never produced diverging or appalling forecasts, further demonstrating its reliability. However, the model's inherent low interpretability and "black box" nature weaken its trustworthiness, thereby undermining its overall reliability.

- **Are the forecasts from the most accurate model applicable as a cornerstone for electricity optimization?**
  Some of the forecasts produced by the LSTM certainly seem adequate to be applied as a cornerstone for electricity optimization, such as the nine-day period depicted in fig. 5.6. However, the model must be adapted to handle control inputs and paired with an optimization algorithm to assess the potential savings more accurately. The model would likely benefit from this adaptation with enhanced accuracy due to the significance of the control inputs. In addition, a confidence measure to make the optimization algorithm able to assess the reliability of the forecast would also be beneficial.

6.2. Further Work

Further work on this topic can broadly be classified into three main categories; further exploration of the HAM paradigm, more comprehensive data collection, and incorporation into an optimization algorithm.

6.2.1. Further Exploration of the HAM Paradigm

Although the CoSTA model implemented in this work did not provide good forecasts, the HAM paradigm contains powerful tools for improving interpretability and trustworthiness, consequently enhancing the reliability of accurate indoor temperature forecasting models. Enhanced reliability will make the model applicable to higher-stakes applications and make it possible to provide some "guarantee" that its forecast is reasonable. Therefore, this paradigm is worth exploring more. This exploration could be undertaken in several ways.

One way would be to rewrite the PBM part of the CoSTA implemented in this work to enable it to consider a broader range of the influencing factors of the temperatures, such as more details on heating loads or a more sophisticated subdivision of the radiation gain from the sun. This would make it easier for the DDM part to determine and generalize the source term. It would also be valuable to investigate how to counteract the non-stationarity of the source term and understand how this influences the model's performance.

Another interesting approach could be to employ different HAM architectures for this forecasting task, for example, Physics-Guided Machine Learning or Data-Driven Equation Discovery. These architectures also contain powerful tools for improving interpretability and trustworthiness and, if successful, enhancing performance.

6.2.2. More Comprehensive Data Collection

To further improve the accuracy of the models, it would be beneficial for the models to include various control inputs, such as radiators, ventilation systems, and fireplaces. These are driving forces of the temperature of the asset and would likely reveal more of the intricate dynamics behind the temperature fluctuations.

At present, the asset of this study does not measure or track any of these control metrics. Therefore, it is necessary to make the data collection more comprehensive by tracking and measuring these control inputs in order to improve the model. In addition, tracking these control inputs would facilitate the models' incorporation into an optimization algorithm, as this optimization algorithm would need forecasts based on different sequences of control input.

6.2.3. Incorporation into an Optimization Algorithm

To truly assess the potential savings attainable from the LSTM model, it would have to be incorporated into an optimization algorithm that optimizes a desired cost function based on its forecasts. This cost function can include different optimization variables, such as energy cost, energy consumption, or a weighted combination of the two.

The LSTM model would then have to be adapted to accommodate control inputs. Once

the model can handle these inputs, it will be able to forecast how the temperature will evolve based on different control input sequences.

Following these modifications, the LSTM model can be incorporated into an optimization algorithm, serving as the foundation for assessing the effect of different control inputs. From this, optimizing the control inputs based on the desired cost function would be possible while reaching the desired temperature at the desired time.

As previously discussed, it would be beneficial for such an optimization algorithm to also implement a confidence measure of the forecast to assess its reliability. Additionally, the algorithm should have a built-in safety measure or fallback mechanism to ensure reaching the desired state at the desired time. This is important in cases where the model's forecast of indoor temperature evolution proves inaccurate.

Bibliography

1. Henrik Larsson Hestnes. Physics-Based Modeling of Building Thermodynamics for Digital Twin. https://www.researchgate.net/publication/371286527_Physics-Based_Modeling_of_Building_Thermodynamics_for_Digital_Twin (Dec. 2022).
2. International Energy Agency. *Buildings* en-GB. Tech. rep. (Sept. 2022). https://www.iea.org/reports/buildings (2023).
3. United States Department of Energy. Chapter 5: Increasing Efficiency of Building Systems and Technologies. en, 39. https://www.energy.gov/sites/prod/files/2017/03/f34/qtr-2015-chapter5.pdf.
4. European Council. *Energy price rise since 2021* en. https://www.consilium.europa.eu/en/infographics/energy-prices-2021/ (2023).
5. Tatiana Mitrova & Akos Losz. *Central Asia's Overlooked Energy Crisis: What It Means for the Global Gas Market* en. Tech. rep. (Columbia University, Mar. 2023). https://www.energypolicy.columbia.edu/central-asias-overlooked-energy-crisis-what-it-means-for-the-global-gas-market/ (2023).
6. International Energy Agency. *Global Energy Crisis* en-GB. Tech. rep. (Feb. 2023). https://www.iea.org/topics/global-energy-crisis (2023).
7. Jean-Pascal Tricoire. *Buildings are the foundation of our energy-efficient future* en. Feb. 2021. https://www.weforum.org/agenda/2021/02/why-the-buildings-of-the-future-are-key-to-an-efficient-energy-ecosystem/ (2023).
8. Nest Labs. Energy Savings from the Nest Learning Thermostat: Energy Bill Analysis Results. https://storage.googleapis.com/nest-public-downloads/press/documents/energy-savings-white-paper.pdf (Feb. 2015).
9. Forburkerrådet. *Spotpriser* nb-NO. https://www.strompris.no/spotpriser (2023).
10. Ketterer, J. C. The impact of wind power generation on the electricity price in Germany. en. *Energy Economics* **44,** 270–280. ISSN: 0140-9883. https://www.sciencedirect.com/science/article/pii/S0140988314000875 (2022) (July 2014).
11. International Energy Agency. *Wind Electricity – Analysis* en-GB. Tech. rep. (Sept. 2022). https://www.iea.org/reports/wind-electricity (2022).
12. Blakseth, S. S., Rasheed, A., Kvamsdal, T. & San, O. Deep neural network enabled corrective source term approach to hybrid analysis and modeling. en. *Neural Networks* **146,** 181–199. ISSN: 0893-6080. https://www.sciencedirect.com/science/article/pii/S0893608021004494 (2022) (Feb. 2022).
13. Blakseth, S. S., Rasheed, A., Kvamsdal, T. & San, O. Combining physics-based and data-driven techniques for reliable hybrid analysis and modeling using the corrective source term approach. en. *Applied Soft Computing* **128,** 109533. ISSN: 1568-4946. https://www.sciencedirect.com/science/article/pii/S156849462200607X (2022) (Oct. 2022).

14. Geneva, N. & Zabaras, N. Transformers for Modeling Physical Systems. *Neural Networks* **146,** 272–289. ISSN: 08936080. http://arxiv.org/abs/2010.03957 (2023) (Feb. 2022).

15. Strogatz, S. H. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering* Second edition. en. OCLC: ocn842877119. ISBN: 978-0-8133-4910-7 (Westview Press, a member of the Perseus Books Group, Boulder, CO, 2015).

16. Jens G. Balchen, Trond Andresen & Bjarne A. Foss. *Reguleringsteknikk* (Institutt for teknisk kybernetikk, 2016).

17. Niloy Ganguly, Andreas Deutsch & Animesh Mukherjee. *Dynamics On and Of Complex Networks* en. https://link.springer.com/book/10.1007/978-0-8176-4751-3 (2022) ().

18. Pawar, S., Ahmed, S. E., San, O. & Rasheed, A. *Hybrid analysis and modeling for next generation of digital twins* tech. rep. arXiv:2101.05908 (arXiv, Jan. 2021). http://arxiv.org/abs/2101.05908 (2022).

19. Doshi-Velez, F. & Kim, B. *Towards A Rigorous Science of Interpretable Machine Learning* tech. rep. arXiv:1702.08608 (arXiv, Mar. 2017). http://arxiv.org/abs/1702.08608 (2022).

20. Amara, F., Agbossou, K., Cardenas, A., Dubé, Y. & Kelouwani, S. Comparison and Simulation of Building Thermal Models for Effective Energy Management. *Smart Grid and Renewable Energy* **06,** 95–112 (Jan. 2015).

21. Aldawi, F. & Alam, F. en. in *Thermofluid Modeling for Energy Efficiency Applications* (eds Khan, M. M. K. & Hassan, N. M. S.) 169–196 (Academic Press, Jan. 2016). ISBN: 978-0-12-802397-6. https://www.sciencedirect.com/science/article/pii/B9780128023976000087 (2023).

22. Jan Vincent Thue. *U-verdi* no. July 2019. https://snl.no/U-verdi (2023).

23. Pilkington. *GLASSFAKTA 2021* https://www.pilkington.com/-/media/pilkington/site-content/norway/glassfakta2021_update1.pdf.

24. Helseth, L. E. *varmekapasitet* no. Mar. 2023. https://snl.no/varmekapasitet (2023).

25. Kämpf, J. H. & Robinson, D. A simplified thermal model to support analysis of urban resource flows. en. *Energy and Buildings* **39,** 445–453. ISSN: 0378-7788. https://www.sciencedirect.com/science/article/pii/S0378778806002192 (2022) (Apr. 2007).

26. Clarke, J. *Energy Simulation in Building Design, Second Edition* 2nd edition. English. ISBN: 978-0-7506-5082-3 (Butterworth-Heinemann, Oxford, Oct. 2001).

27. Maharana, K., Mondal, S. & Nemade, B. A review: Data pre-processing and data augmentation techniques. en. *Global Transitions Proceedings. International Conference on Intelligent Engineering Approach(ICIEA-2022)* **3,** 91–99. ISSN: 2666-285X. https://www.sciencedirect.com/science/article/pii/S2666285X22000565 (2023) (June 2022).

28. Nguyen, M. *et al. Predicting Alzheimer's disease progression using deep recurrent neural networks* (Sept. 2019).

29. Sola, J. & Sevilla, J. Importance of input data normalization for the application of neural networks to complex industrial problems. *Nuclear Science, IEEE Transactions on* **44,** 1464–1468 (July 1997).

30. Singh, D. & Singh, B. Investigating the impact of data normalization on classification performance. en. *Applied Soft Computing* **97,** 105524. ISSN: 1568-4946. https://www.sciencedirect.com/science/article/pii/S1568494619302947 (2023) (Dec. 2020).

31. Kappal, S. *Data Normalization Using Median & Median Absolute Deviation (MMAD) based Z-Score for Robust Predictions vs. Min-Max Normalization* (June 2019).

32. Iglewicz, B. & Hoaglin, D. C. *How to Detect and Handle Outliers* 1st edition. English. ISBN: 978-0-87389-247-6 (Asq Pr, Milwaukee, Wis, Jan. 1993).

33. Hota, H., Handa, R. & Shrivas, A. *Time Series Data Prediction Using Sliding Window Based RBF Neural Network* in (2017). https://www.semanticscholar.org/paper/Time-Series-Data-Prediction-Using-Sliding-Window-Hota-Handa/91037f01fd4b845eadca0b53f5dc00d9f61ac493 (2023).

34. Bao, A., Gildin, E., Huang, J. & Coutinho, E. *Data-Driven End-To-End Production Prediction of Oil Reservoirs by EnKF-Enhanced Recurrent Neural Networks* (July 2020).

35. Abdul Salam, M., Taher, A., Elgendy, M. & Mohamed, K. The Effect of Different Dimensionality Reduction Techniques on Machine Learning Overfitting Problem. *International Journal of Advanced Computer Science and Applications* **12** (Jan. 2021).

36. Bellman, R. Dynamic programming. eng. *Science (New York, N.Y.)* **153,** 34–37. ISSN: 0036-8075 (July 1966).

37. Hall, M. A. Correlation-based Feature Selection for Machine Learning. en (Apr. 1999).

38. Iyer, V. & Roy Chowdhury, K. Spectral Analysis: Time Series Analysis in Frequency Domain. *The IUP Journal of Applied Economics* **VIII,** 83–101 (Jan. 2009).

39. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* Illustrated edition. English. ISBN: 978-0-262-03561-3 (The MIT Press, Cambridge, Massachusetts, Nov. 2016).

40. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* **15,** 1929–1958 (June 2014).

41. Prechelt, L. en. in *Neural Networks: Tricks of the Trade* (eds Orr, G. B. & Müller, K.-R.) 55–69 (Springer, Berlin, Heidelberg, 1998). ISBN: 978-3-540-49430-0. https://doi.org/10.1007/3-540-49430-8_3 (2023).

42. Lundby, E., Rasheed, A., Halvorsen, I. & Gravdahl, J. *Sparse deep neural networks for modeling aluminum electrolysis dynamics* (Sept. 2022).

43. Hochreiter, S. & Schmidhuber, J. Long Short-term Memory. *Neural computation* **9,** 1735–80 (Dec. 1997).

44. Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* **5.** Conference Name: IEEE Transactions on Neural Networks, 157–166. ISSN: 1941-0093 (Mar. 1994).

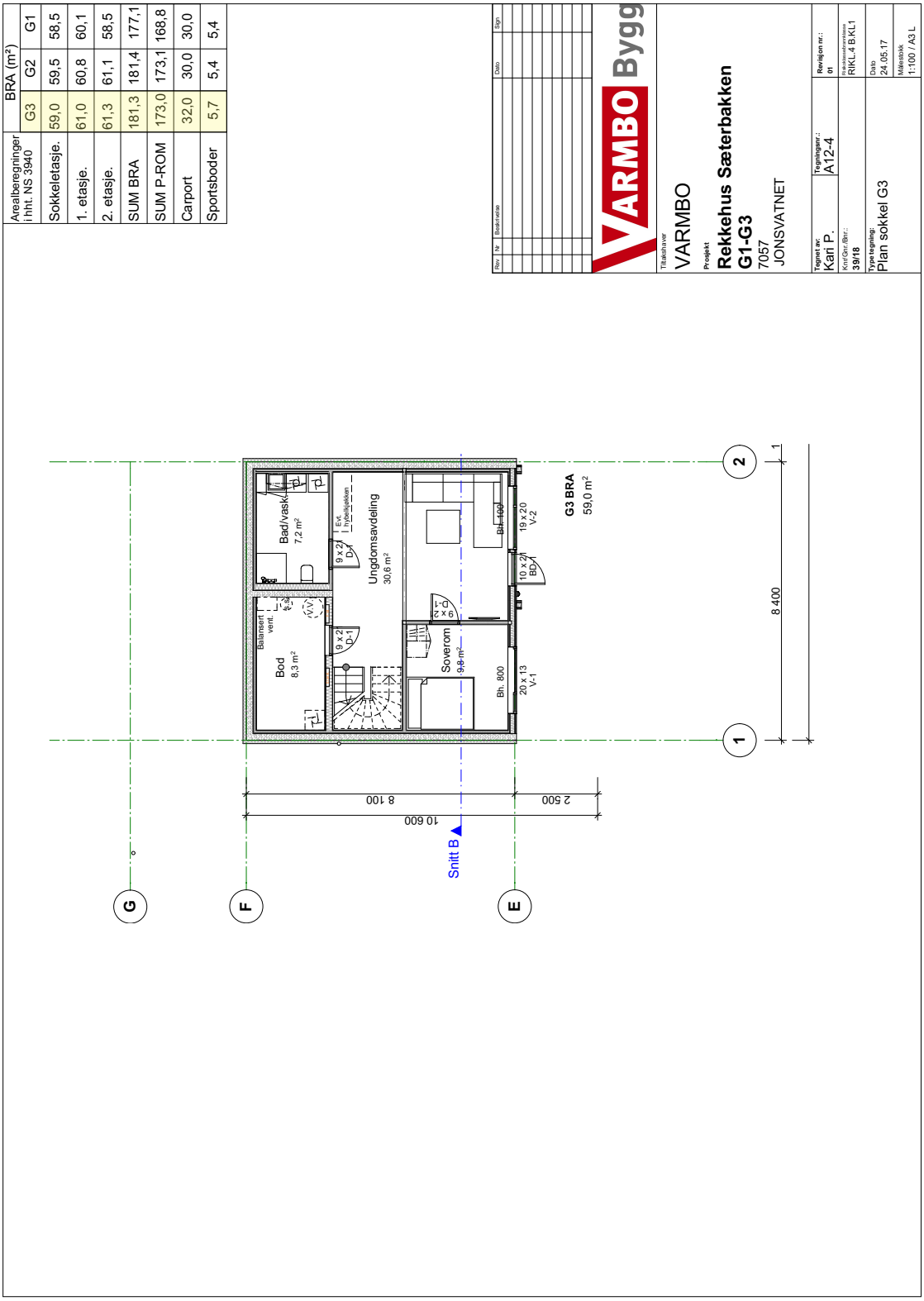
45. Van Houdt, G., Mosquera, C. & Nápoles, G. *A Review on the Long Short-Term Memory Model* tech. rep. (Dec. 2020).
46. Vaswani, A. *et al. Attention Is All You Need* tech. rep. (arXiv, Dec. 2017). http://arxiv.org/abs/1706.03762 (2023).
47. Lin, T., Wang, Y., Liu, X. & Qiu, X. *A Survey of Transformers* tech. rep. (arXiv, June 2021). http://arxiv.org/abs/2106.04554 (2023).
48. Bahrini, A. *et al. ChatGPT: Applications, Opportunities, and Threats* tech. rep. arXiv:2304.09103 (arXiv, Apr. 2023). http://arxiv.org/abs/2304.09103 (2023).
49. Pope, R. *et al. Efficiently Scaling Transformer Inference* tech. rep. arXiv:2211.05102 (arXiv, Nov. 2022). http://arxiv.org/abs/2211.05102 (2023).
50. Zuo, S., Jiang, H., Li, Z., Zhao, T. & Zha, H. *Transformer Hawkes Process* tech. rep. arXiv:2002.09291 (arXiv, Feb. 2021). http://arxiv.org/abs/2002.09291 (2023).
51. San, O., Rasheed, A. & Kvamsdal, T. Hybrid analysis and modeling, eclecticism, and multifidelity computing toward digital twin revolution. en. *GAMM-Mitteilungen* **44,** e202100007. ISSN: 1522-2608. https://onlinelibrary.wiley.com/doi/abs/10.1002/gamm.202100007 (2023) (2021).
52. *PyTorch* en. https://www.pytorch.org (2023).
53. *Autograd — PyTorch Tutorials 2.0.1+cu117 documentation* https://pytorch.org/tutorials/beginner/former_torchies/autograd_tutorial_old.html#sphx-glr-beginner-former-torchies-autograd-tutorial-old-py (2023).
54. Kotsiantis, S., Kanellopoulos, D. & Pintelas, P. Data Preprocessing for Supervised Learning. *International Journal of Computer Science* **1,** 111–117 (Jan. 2006).
55. Guyon, I. & Elisseeff, A. An introduction to variable and feature selection. *The Journal of Machine Learning Research* **3,** 1157–1182. ISSN: 1532-4435 (Mar. 2003).
56. Hornik, K. Approximation capabilities of multilayer feedforward networks. en. *Neural Networks* **4,** 251–257. ISSN: 0893-6080. https://www.sciencedirect.com/science/article/pii/089360809190009T (2023) (Jan. 1991).
57. Dixit, A. & Jain, S. *Effect of stationarity on traditional machine learning models: Time series analysis* en. in *2021 Thirteenth International Conference on Contemporary Computing (IC3-2021)* (ACM, Noida India, Aug. 2021), 303–308. ISBN: 978-1-4503-8920-4. https://dl.acm.org/doi/10.1145/3474124.3474167 (2023).
58. *sklearn.preprocessing.StandardScaler* en. https://scikit-learn/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (2023).
59. Yu, T. & Zhu, H. *Hyper-Parameter Optimization: A Review of Algorithms and Applications* tech. rep. arXiv:2003.05689 (arXiv, Mar. 2020). http://arxiv.org/abs/2003.05689 (2023).
60. Sivle, A. D. & Evans, A. K. D. *værvarsling* no. Sept. 2022. https://snl.no/v%C3%A6rvarsling (2023).
61. Norsk klimaservicesenter. *Observasjoner og værstatistikk* Jan. 2023. https://seklima.met.no/observations/ (2023).
62. Norwegian Building Authority. *Byggteknisk forskrift (TEK17)* no. https://dibk.no/regelverk/byggteknisk-forskrift-tek17/ (2022).

A. Appendix

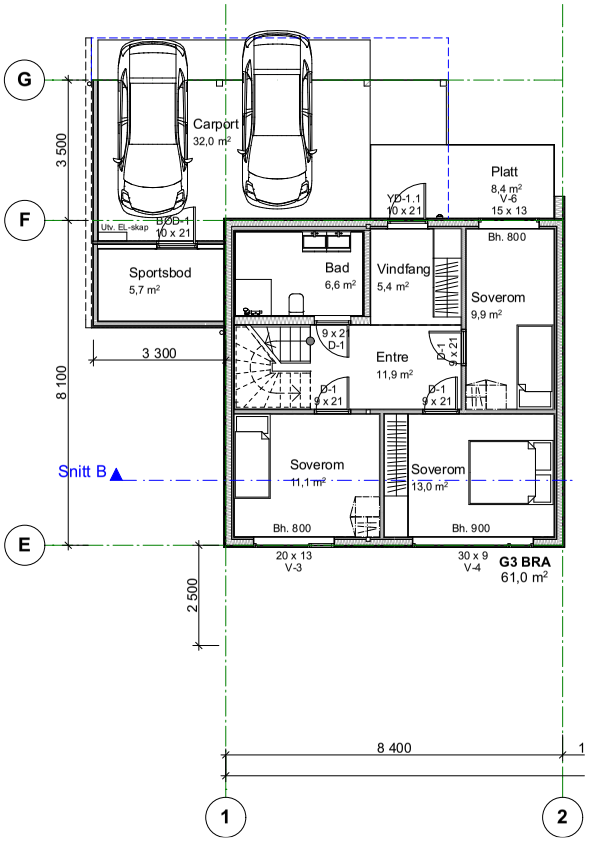
A.1. Asset Specifications

A.1.1. Floor Plan

Arealberegninger i hht. NS 3940	BRA (m²)		
	G3	G2	G1
Sokkeletasje.	59,0	59,5	58,5
1. etasje.	61,0	60,8	60,1
2. etasje.	61,3	61,1	58,5
SUM BRA	181,3	181,4	177,1
SUM P-ROM	173,0	173,1	168,8
Carport	32,0	30,0	30,0
Sportsboder	5,7	5,4	5,4



Arealberegninger i hht. NS 3940	BRA (m²)		
	G3	G2	G1
Sokkeletasje.	59,0	59,5	58,5
1. etasje.	61,0	60,8	60,1
2. etasje.	61,3	61,1	58,5
SUM BRA	181,3	181,4	177,1
SUM P-ROM	173,0	173,1	168,8
Carport	32,0	30,0	30,0
Sportsboder	5,7	5,4	5,4



Rev	Nr	Beskrivelse	Dato	Sign

VARMBO Bygg

Tiltakshaver

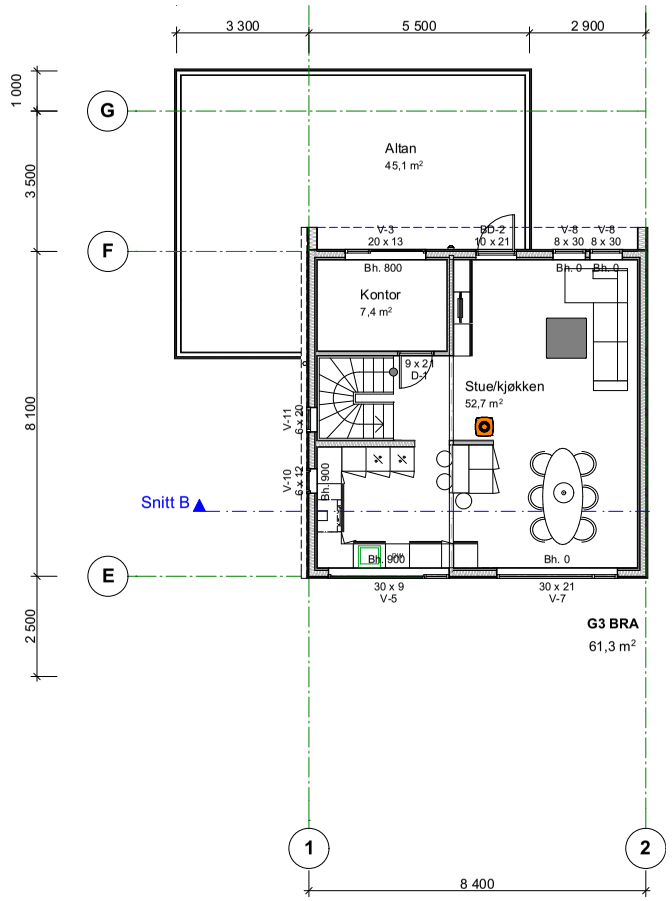
VARMBO

Prosjekt

Rekkehus Sæterbakken
G1-G3
7057
JONSVATNET

Tegnet av:	Tegningsnr.:	Revisjon nr.:
Kari P.	A12-5	01
Knr/Gnr./Bnr.: 39/18		Rekkeklassebrannklasse: RIKL.4 B.KL1
Type tegning: Plan 1.etg G3		Dato: 24.05.17
		Målestokk: 1:100 / A3 L

Arealberegninger i hht. NS 3940	BRA (m²)		
	G3	G2	G1
Sokkeletasje.	59,0	59,5	58,5
1. etasje.	61,0	60,8	60,1
2. etasje.	61,3	61,1	58,5
SUM BRA	181,3	181,4	177,1
SUM P-ROM	173,0	173,1	168,8
Carport	32,0	30,0	30,0
Sportsboder	5,7	5,4	5,4



Rev	Nr	Beskrivelse	Dato	Sign

VARMBO Bygg

Tiltakshaver

VARMBO

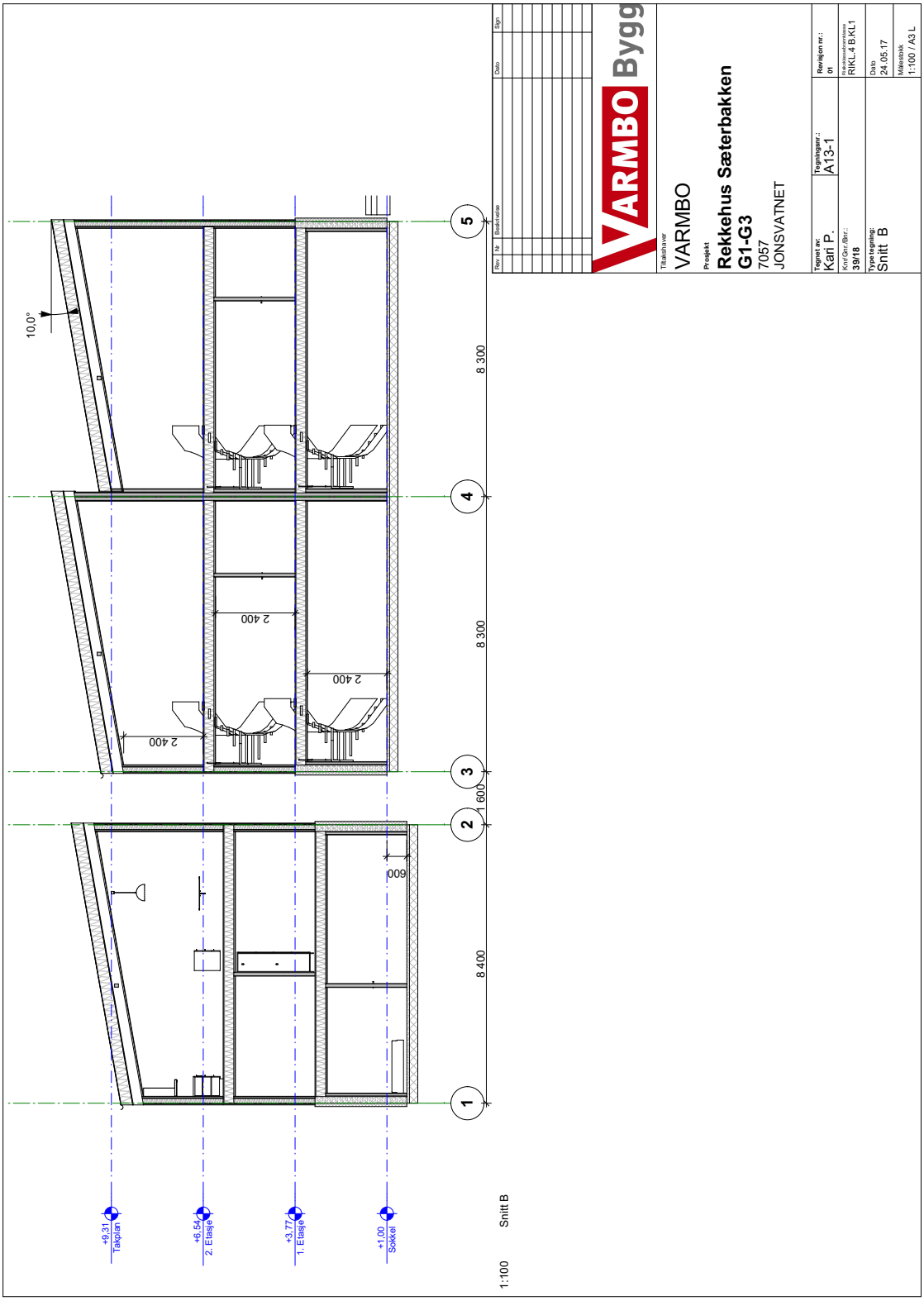
Prosjekt

Rekkehus Sæterbakken
G1-G3
7057
JONSVATNET

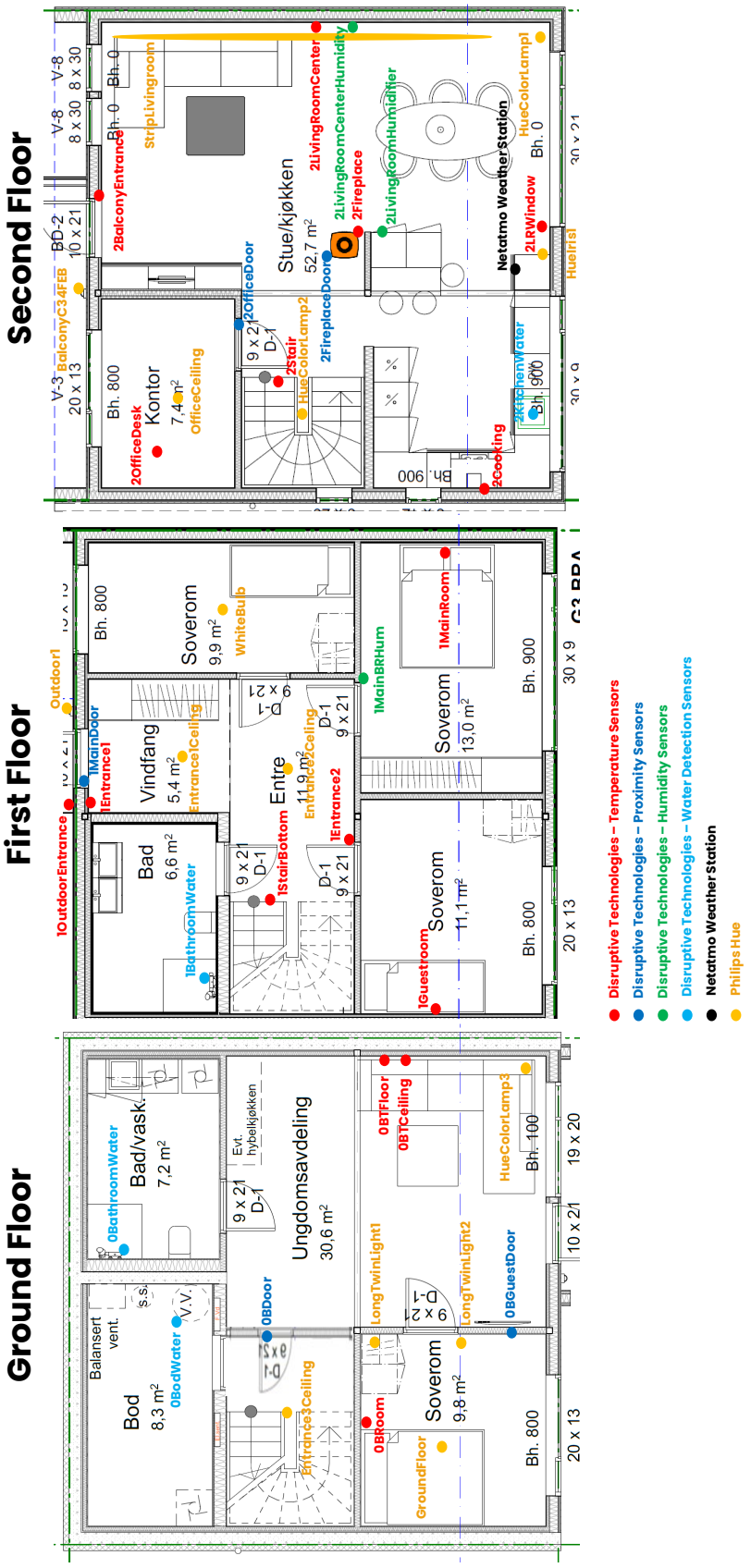
Tegnet av:	Tegningsnr.:	Revisjon nr.:
Kari P.	A12-6	01
Knr/Gnr./Bnr.: 39/18		Rekkeklassebrannklasse RIKL.4 B.KL1
Type tegning: Plan 2.etg G3		Dato 24.05.17
		Målestokk 1:100 / A3 L

A.1.2. Section Plan

The asset of the study is the leftmost of the three houses depicted below.



A.1.3. Floor Plan with Sensor Layout



A.1.4. Balanced Ventilation Datasheet

Calculation report

Flexit Nordic S 4 R E 800 L 230

Version 2022.1.1.182
16.11.2022
NO

Project

Name	Stokkasen51
Project ID	
Country	NO
Location	Oslo
Customer	
Designed by	adil rasheed
Information	

Assumptions

Atmospheric pressure 101325 Pa
Air density 1.2 kg/m³
Sound duct in accordance with ISO 5136
Ambient sound in accordance with ISO 9614-2

Unit specification

Flexit participates in the ECP programme for RAHU.
Check ongoing validity of certificate:
www.eurovent-certification.com

Description	
Information	
Weight	62 kg
Fuse size	230V 10A
Unit version	Left
General	
Comparison winter temperature (DUTv)	-19.8 °C
Indoor temperature	21 °C
Desired supply air temperature	18 °C
Supply winter	
Airflow winter	0.08 m³/s
Pressure drop winter	100 Pa
Extract winter	
Airflow winter	0.08 m³/s
Pressure drop winter	100 Pa
Energy Result	
Air heating coil max power requirement	0.29 kW
Ventilation demand without heat recovery	9470 kWh/year
Energy used supply fan	428 kWh/year
Energy used extract fan	411 kWh/year
Energy requirement after heat exchanger	177 kWh/year
Total energy used	1016 kWh/year
Energy saved	8453 kWh/year
Yearly energy efficiency	89 %

Performance

Ambient sound (Lw)	41 dB(A)
Ambient sound (Lp) with room attenuation	37 dB(A)
Room attenuation	4 dB
Temperature efficiency (EN308)	85 %
Temperature efficiency (EN13141-7)	85 %
SFP total winter	1.2 kJ/m³

Technical specification

Filter, supply

Filter class	ePM1 55%
Dimensions	365x247x31
Number of filters	1 pcs
Area	1.06 m²
Winter	
Dimension pressure drop	71 Pa
Start pressure drop	59 Pa
End pressure drop	250 Pa

Filter, extract

Filter class	ePM1 55%
Dimensions	365x247x31
Number of filters	1 pcs
Area	1.06 m²
Winter	
Dimension pressure drop	71 Pa
Start pressure drop	59 Pa
End pressure drop	250 Pa

Hex

Type	ST1-1.6-200-Ø350
Winter	
Temperature outdoor	-19.8 °C
Relative humidity outdoor	91 %
Temperature supply	15 °C
Relative humidity supply	35 %
Pressure drop supply	106 Pa
Temperature efficiency supply (EN308)	85 %
Humidity efficiency supply	78 %
Temperature extract	21 °C
Relative humidity extract	30 %
Temperature exhaust	-13.8 °C
Relative humidity exhaust	99 %
Pressure drop exhaust	106 Pa
Humidity efficiency exhaust	88 %
Energy Calculation	
Power usage	4.1 W

Fan, supply

Winter	
Adjustment	67 %
Power	49 W
Current	0.36 A
Power factor	0.576
SFP	0.6 kJ/m³
RPM	2617 rpm

Fan, extract

Winter	
Adjustment	65 %
Power	47 W
Current	0.35 A

Power factor	0.583
SFP	0.6 kJ/m³
RPM	2592 rpm

After heater, electrical

Max effect	0.8 kW
Winter	
Effect	0.3 kW
Temperature before	15 °C
Temperature after	18 °C

Sound data, supply

Frequency band	Hz	63	125	250	500	1k	2k	4k	8k	Lw	Total Lw(A)
Supply duct, winter		67	67	64	59	55	54	45	39	dB	62 dB(A)

Sound data, extract

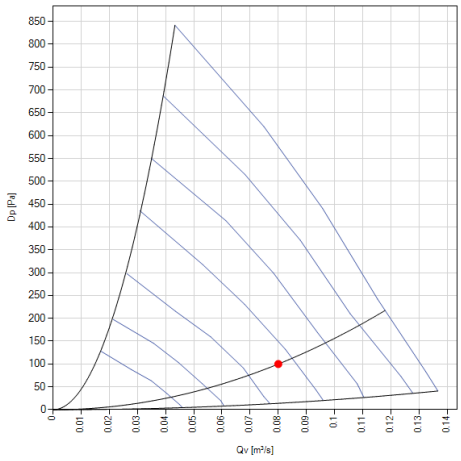
Frequency band	Hz	63	125	250	500	1k	2k	4k	8k	Lw	Total Lw(A)
Extract duct, winter		58	59	56	46	38	35	26	25	dB	50 dB(A)

Sound data, ambient

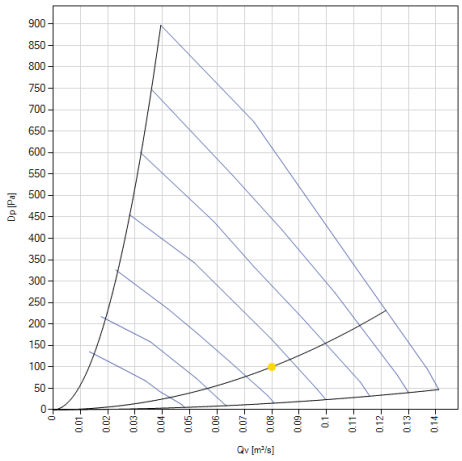
Frequency band	Hz	63	125	250	500	1k	2k	4k	8k	Lw	Total Lw(A)
Ambient, winter		54	49	44	35	30	33	29	26	dB	41 dB(A)
Ambient sound (Lp) with room attenuation 4dB											37 dB(A)

Diagrams

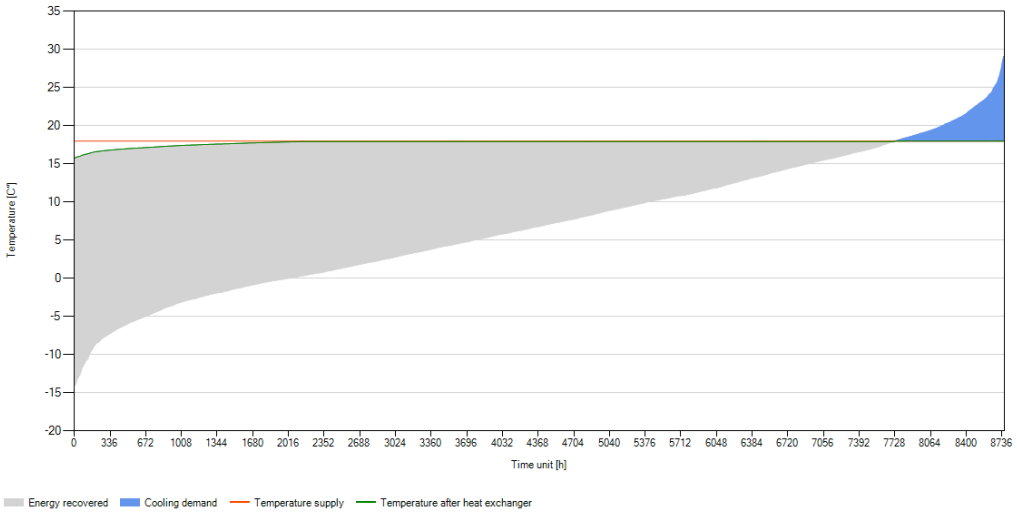
Supply



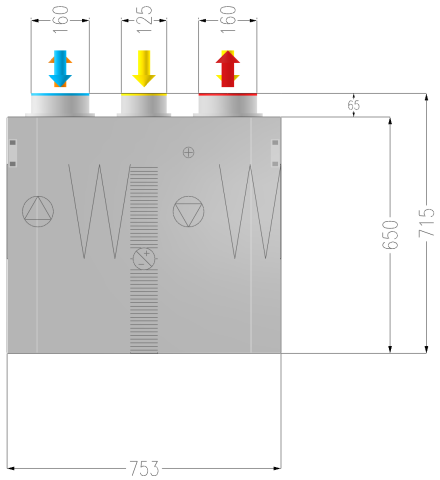
Extract

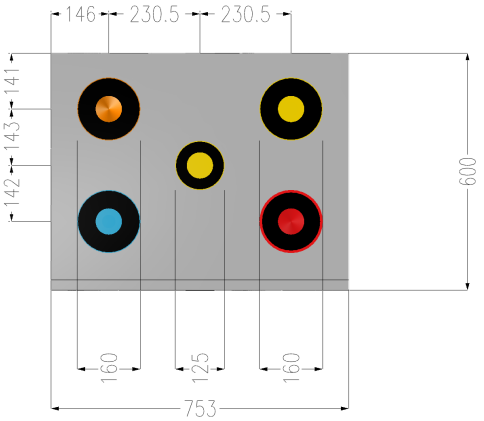


Ventilation duration diagram



Dimensions

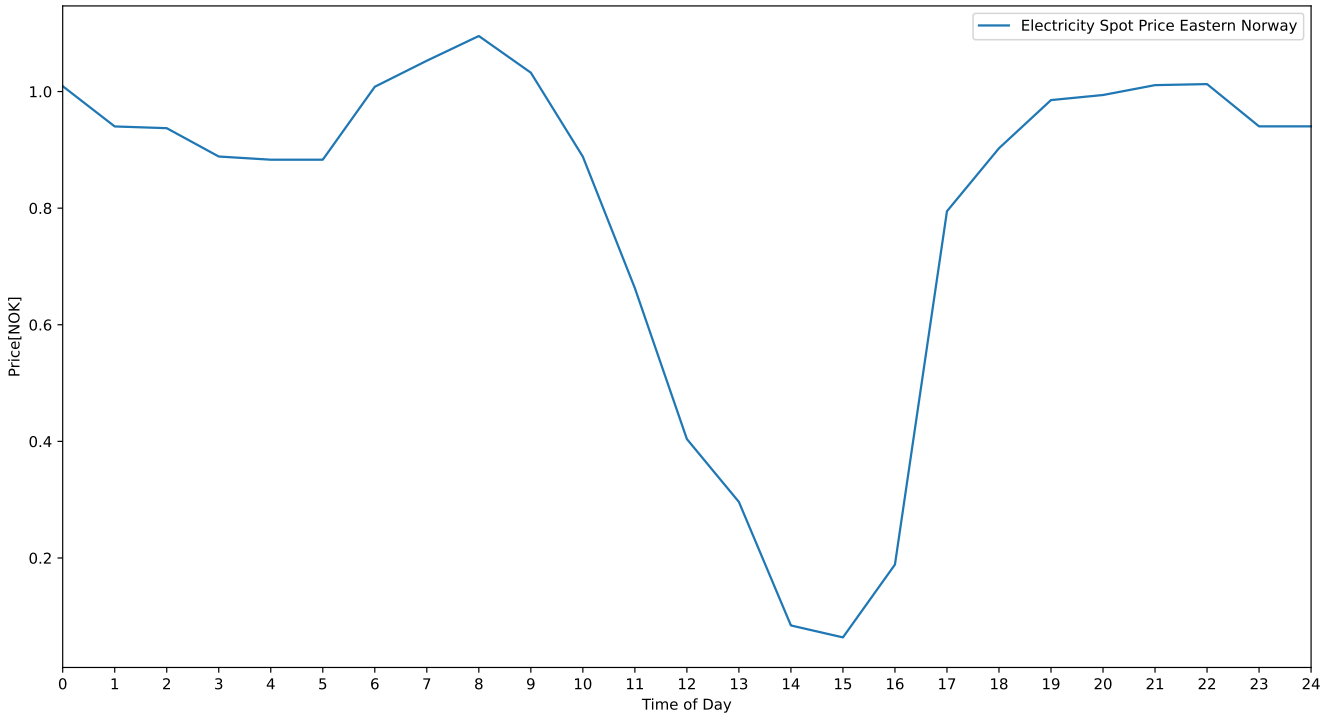


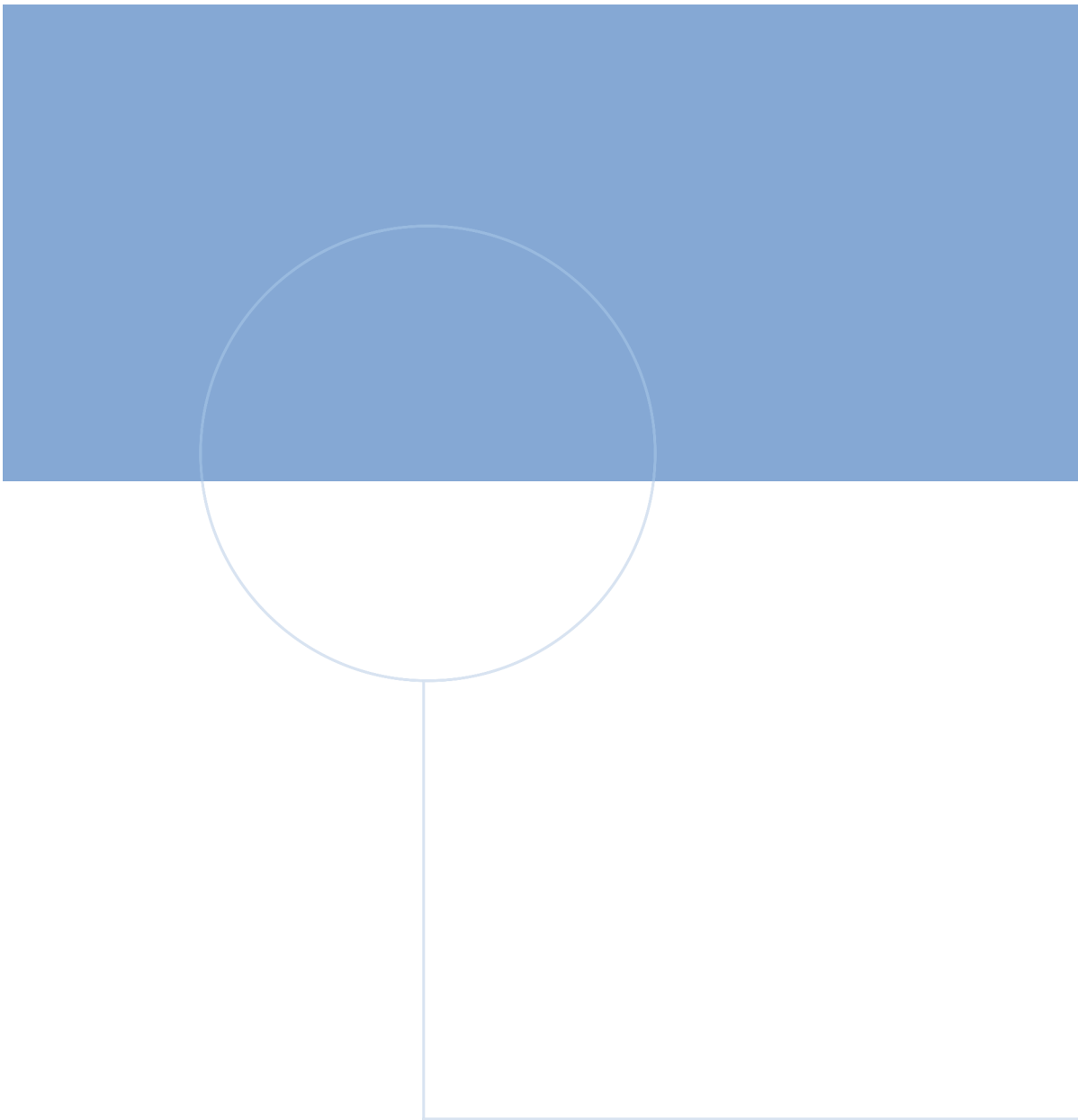


Flexit participates in the ECP programme for RAHU.
Check ongoing validity of certificate:
www.eurovent-certification.com

A.2. Electricity Spot Price Eastern Norway 23rd of May 2023

The electricity spot price evolution for eastern Norway on the 23rd of May, 2023, is presented below. The data was collected from Strømpris.no, a service provided by Forbrukerrådet[9].





NTNU

Norwegian University of
Science and Technology