Advancing Building Energy Efficiency with Physics-Informed Neural Networks for Time Series Forecasting

Seyedamin Mohimanianpour

Abstract

The rapid urbanization and population growth have driven buildings to consume a significant portion of the world's energy, with expectations of further increases in the future. Governments have responded by implementing measures such as raising energy prices and enforcing regulations for constructing energy-efficient buildings. Considering these efforts, advanced control techniques, including Model Predictive Control (MPC), have emerged as promising solutions for reducing energy usage. However, conventional MPC models face challenges due to their reliance on costly physical models. In contrast, data-driven approaches, leveraging historical building data to enhance energy efficiency and indoor comfort. Despite their potential, these approaches face challenges such as high data requirements and interpretability issues. Physics-informed Neural Networks (PINN) bridge the gap between physics-based modeling and data-driven approaches, integrating physical principles into machine learning frameworks to improve prediction accuracy and robustness. This thesis explores the application of PINN and other machine learning methods for time series forecasting of energy consumption in buildings, providing insights into their effectiveness and limitations. Through a comparative analysis, it demonstrates the data efficiency of PINN and its potential for enhancing energy management strategies in buildings.

Sammendrag

Rask urbanisering og befolkningsvekst har drevet bygninger til å forbruke en betydelig del av verdens energi, med forventninger om ytterligere økninger i fremtiden. Regjeringer har svart ved å implementere tiltak som å øke energiprisene og håndheve reguleringer for å bygge energieffektive bygninger. Med tanke på disse innsatsene har avanserte kontrollteknikker, inkludert Model Pre- dictive Control (MPC), blitt fremhevet som lovende løsninger for å redusere energiforbruket. Imidlertid står konvensjonelle MPC-modeller overfor utfordringer på grunn av deres avhengighet av kostbare fysiske modeller. I motsetning til dette, benytter datadrevne tilnærminger historiske bygningsdata for å forbedre energieffektiviteten og innendørs komfort. Til tross for deres potensial, står slike tilnærminger overfor utfordringer som høye datakrav og tolkningsproblemer. Physics-informed Neural Networks (PINNs) brobygger gapet mellom fysikkbasert modellering og datadrevne tilnærminger, og integrerer fysiske prinsipper i maskinlæringsrammeverk for å forbedre prediksjons nøyaktighet og robusthet. Denne avhandlingen utforsker bruken av PINN og andre maskinlæringsmetoder for tidsrekkeprognoser av energiforbruk i bygninger, og gir innsikt i deres effektivitet og begrensninger. Gjennom en sammenlignende analyse demonstrerer den dataeffektiviteten til PINN og potensialet til å forbedre energiforvaltningsstrategier i bygninger.

Contents

Figures

Tables

Acronyms

Q_1 First quartile. 13

Q_3 Third quartile. 13

2R2C 2-state resistance-capacitance network model. 5, 12, 13

ACF Auto-correlation function. ix, 18, 31, 32

ADF Augmented Dickey-Fuller. 17

ANN Artificial Neural Networks. 1

AR Auto Regression. 19

ARIMA Autoregressive Integrated Moving Average. 17–20, 31

BPTT Backpropagation through time. 24

CNN convolutional Neural Networks. 1

D Difference Component. 19

I Integration. 19

IQR interquartile range. 13

LSTM Long Short-Term Memory. vii, 1–3, 22–24, 35, 36, 39, 41, 42

m number of observations per season. 19

MA Moving Average. 19

MAE Mean Absolute Error. 14, 39

MDP Markov Decision Process. 6, 8

MLP Multilayer Perceptron. 27

MPC Model Predictive Control. 1, 2

MSE Mean Squared Error. 14, 15, 35

NNs Neural Networks. 9

P Auto-regressive Component. 19

PACF Partial auto-correlation function. ix, 18, 19, 31, 32

PFM Prophet Forecasting Model. vii, 2, 3, 21, 32

PINN Physics-Informed Neural Network. vii, ix, 3, 9, 14, 26, 34, 39, 41, 42

Q Moving Average Component. 19

RF Random Forest. 1

RMSE Root Mean Squared Error. 14, 15

RNN Recurrent Neural Network. 2, 22, 41

SARIMA Seasonal Autoregressive Integrated Moving Average. vii, ix, 2, 17–20, 30, 31, 33, 39, 41, 42

SGD Stochastic Gradient Descent. 7

SVR Support Vector Regression. 1

Chapter 1

Introduction

Nowadays, the process of urbanization coupled with continuously growing populations has led to buildings consuming approximately 40 % of the world's energy production, with expectations of this consumption rate rising even further in the near future. The government has implemented various measures to address the growing demand for energy. For instance, they have raised energy prices and enforced stringent regulations for constructing more energy-efficient buildings. In addition to costly measures like upgrading heating and cooling systems and improving a building's structure, advanced control techniques offer an alternative for reducing energy usage. Model Predictive Control (MPC) is one such method [1]. It involves utilizing a mathematical model of system dynamics to optimize control actions based on cost considerations and comfort requirements within a predictive timeframe.

In simulated scenarios [2–5] and real-world building settings [6–9], Model Predictive Control (MPC) has proven to be an effective application, yielding notable energy savings over traditional control techniques. The conventional models that are employed in the construction of Model Predictive Control (MPC) often depend on physical concepts, like heat transfer and thermodynamics, as mentioned in [6–9]. It is possible to build these models using theoretical frameworks, expensive excitation experiments, or a combination of both. It can be difficult to justify investing in these models, though, because their creation and maintenance are typically thought to be excessively costly. As such, this barrier might prevent MPC from being widely used for commercial purposes in building environments.

The data-driven approaches are quickly growing because they avoid the shortcomings of traditional RC models by using only building historical data, hence addressing the aforementioned constraints of conventional RC models. In addition to deep learning techniques like Long Short-Term Memory (LSTM), convolutional Neural Networks (CNN), and Artificial Neural Networks (ANN), the often-used learning-based model includes traditional machine learning techniques like Support Vector Regression (SVR), Random Forest (RF), and XGBoost [10, 11]. Because of the rapid development of smart buildings, which produce vast amounts of data throughout the course of their lives, these techniques have garnered significant attention. A learning based MPC for real-time building control was presented by [12]. In addition to increased energy efficiency, there was a noticeable improvement in indoor thermal comfort. However because this model's performance is heavily dependent on the caliber and quantity of historical data, it cannot be used for buildings that are still in the design stage or lack sufficient important data. Since this model only uses historical data, the quantity and quality of the data are crucial. In [13], data-driven models were developed using data spanning weeks, months, and years. As fresh data is collected and accumulated over time, the model's parameters can be continuously changed for improved prediction performance. In addition to the size of the training dataset, researchers have also looked into the importance of the prediction horizon length, considering factors like computational cost, prediction accuracy, and control performance [14, 15]. [16] utilized learning-based Model Predictive Control (MPC) to forecast indoor temperature and supply/return water temperature for residential AC systems, exploring various prediction horizons. Furthermore, methods like autoencoder, generative adversarial networks, and transformations are employed to enhance model robustness and prediction accuracy when dealing with limited data. Despite their ease of development and deployment in building control systems, these learning-based models still face challenges such as high data requirements, interpretability issues, and limitations in generalization ability.

In forecasting the time series data of energy consumption in a building, several methods can be employed, each with its strengths and limitations. Seasonal Autoregressive Integrated Moving Average (SARIMA) models are a classical statistical approach widely used for time series forecasting. SARIMA models are effective at capturing seasonal patterns and trends in the data by incorporating autoregressive, differencing, and moving average components. By explicitly modeling seasonal differences and autocorrelation, SARIMA models can provide good forecasts for energy consumption in buildings with recurring seasonal variations [17–21].

Prophet Forecasting Model is another time series forecasting method. Utilizing the benefits of Prophet's adaptable and user-friendly framework enhances the accessibility and precision of energy consumption forecasting. By incorporating seasonality, holidays, and trend components, Prophet can capture complex patterns inherent in energy consumption data, making it well-suited for forecasting tasks. Moreover, Prophet's ability to handle missing data and outliers, along with its customizable parameters for uncertainty estimation, further enhances its utility in energy consumption prediction. Additionally, Prophet's scalability and ease of implementation make it particularly attractive for practical applications in building energy management systems [22–26] With its proven performance and user-friendly interface, Prophet stands as a valuable asset in the arsenal of tools for accurately forecasting energy consumption in buildings, facilitating informed decision-making and efficient energy management strategies.

On the other hand, Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) architecture specifically designed to capture long-term dependencies and temporal dynamics in sequential data. LSTM networks excel in capturing complex patterns and temporal relationships in time series data, making them well-suited for energy consumption forecasting in buildings. LSTM networks can effectively capture dynamic changes in energy consumption over time, such as daily and seasonal variations, occupancy patterns, and weather conditions [18, 27–30]. Moreover, traditional feedforward neural networks, also known as regular neural networks, offer another approach to time series forecasting. These networks learn complex relationships between input features and energy consumption through multiple hidden layers of neurons. While not explicitly designed for temporal modeling like LSTM networks, regular neural networks can still capture nonlinear patterns and dependencies within the data. However, they may struggle to capture long-term dependencies and temporal dynamics as effectively as LSTM networks, particularly in scenarios with noisy or sparse data [31–36].

Physics-Informed Neural Network (PINN) represents a powerful paradigm that bridges the gap between traditional physics-based modeling and data-driven approaches. In PINN, knowledge of the underlying physical principles governing a system is integrated into the machine learning framework, enabling more accurate and robust predictions, particularly in scenarios with limited or noisy data. By incorporating known physical laws, constraints, and relationships as a part of the learning process, PINN methods enhance model interpretability, generalization, and transferability. These techniques find widespread applications across various domains, including fluid dynamics, climate modeling, materials science, and engineering [37–44], offering valuable insights into complex phenomena while leveraging the efficiency and scalability of modern machine learning algorithms.

The purpose of this thesis can be outlined as follows:

- Using real-world data on energy consumption of a building located in the south of Norway, two physics-informed neural networks are applied in making predictions over longer periods.
- A comparative analysis between the PINN and other machine learning methods applied for time series predictions.
- Moreover, it is illustrated that training these physics-informed neural network architectures is a more data-efficient process, requiring less training data compared to conventional neural networks without physics-based constraints.

First, the theoretical foundations is considered in "Section 2: Mathematical Modeling Section", explaining the thermal model of a building and the mathematical framework for modeling. The core of this thesis lies on "Physics-Informed Neural Network (PINN)" methodologies, explored in detail in Chapter 3. In Chapter 4, alternative machine learning methods such as SARIMA, PFM, and LSTM are applied. In Chapter 5, "Results and Discussions," significant insights obtained from real-world building datasets are presented, detailing the data cleaning procedures and the results derived from each predictive model. The key findings drawn from the preceding chapters are brought together and summarized in Chapter 6.

Chapter 2

Mathematical Modeling

2.1 Thermal Model of a Building

The 2-state resistance-capacitance network model (2R2C) [45] is being utilized for modeling of thermal behavior of a building, Figure 2.1.

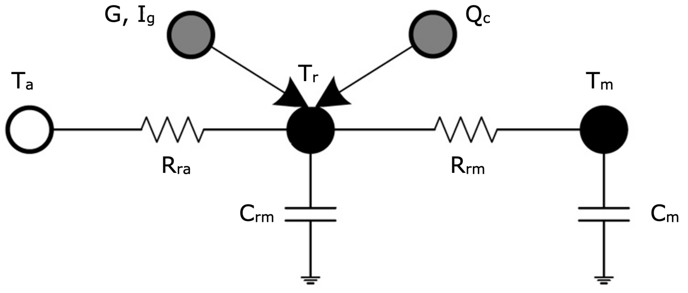


Figure 2.1: The resistance-capacitance network of the building thermal model

The cooling capacity, Q_c , for a building is expressed as

$$Q_c = \dot{m} C_p \left(T_s - T_r\right) \tag{2.1}$$

defined as

$$\begin{aligned}\begin{bmatrix} \dot{T}_\mathrm{r} \\ \dot{T}_\mathrm{m} \end{bmatrix} = &\begin{bmatrix} -\left(\frac{1}{C_\mathrm{r} R_\mathrm{ra}} + \frac{1}{C_\mathrm{r} R_\mathrm{rm}}\right) & \frac{1}{C_\mathrm{r} R_\mathrm{rm}} \\ \frac{1}{C_\mathrm{m} R_\mathrm{rm}} & -\frac{1}{C_\mathrm{m} R_\mathrm{rm}} \end{bmatrix} \cdot \begin{bmatrix} T_\mathrm{r} \\ T_\mathrm{m} \end{bmatrix} + \begin{bmatrix} \frac{C_\mathrm{p} T_\mathrm{s}}{C_\mathrm{r}} \\ 0 \end{bmatrix} \dot{m} \\ &+ \begin{bmatrix} -\frac{C_\mathrm{p}}{C_\mathrm{r}} & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} T_\mathrm{r} \\ T_\mathrm{m} \end{bmatrix} \dot{m} + \begin{bmatrix} \frac{1}{C_\mathrm{r} R_\mathrm{ra}} & \frac{\gamma}{C_\mathrm{r}} & \frac{1}{C_\mathrm{r}} \\ 0 & \frac{1-\alpha}{C_m} & \frac{1-\beta}{C_m} \end{bmatrix} \cdot \begin{bmatrix} T_\mathrm{a} \\ G \\ I_\mathrm{g} \end{bmatrix}.\end{aligned} \tag{2.2}$$

where T_m is the temperature of the building's lumped thermal mass; C_r and C_m are the thermal capacitances of the room and the thermal mass respectively; T_a is defined as the ambient temperature; R_{ra} and R_{rm} are the thermal resistances between the room and the ambient, and between the room and the thermal mass respectively; G, γ and I_g are designated as the solar irradiance, the solar irradiance absorption factor and the internal heat gain, respectively. It is worth noting that T_m represents a hidden state of the system that cannot be directly observed and, in many instances, is notably challenging to approximate. Consequently, this modeling strategy results in a partially observable model of the building. Given Eq. (2.1), Eq. (2.2) is simplified as [46],

$$\begin{bmatrix} T_\mathrm{r} \\ T_\mathrm{m} \end{bmatrix} = \begin{bmatrix} -\left(\frac{1}{C_\mathrm{r}R_\mathrm{rm}}+\frac{1}{C_\mathrm{r}R_\mathrm{rm}}\right) & \frac{1}{C_\mathrm{r}R_\mathrm{rm}} \\ \frac{1}{C_\mathrm{m}R_\mathrm{rm}} & -\frac{1}{C_\mathrm{m}R_\mathrm{rm}} \end{bmatrix} \begin{bmatrix} T_\mathrm{r} \\ T_\mathrm{m} \end{bmatrix} + \begin{bmatrix} \frac{1}{C_\mathrm{r}} \\ 0 \end{bmatrix} Q_e + \begin{bmatrix} \frac{1}{C_\mathrm{r}R_\mathrm{ra}} & \frac{\gamma}{C_\mathrm{r}} & \frac{1}{C_\mathrm{r}} \\ 0 & \frac{1-\alpha}{C_\mathrm{m}} & \frac{1-\beta}{C_\mathrm{m}} \end{bmatrix} \begin{bmatrix} T_\mathrm{m} \\ G \\ I_\mathrm{g} \end{bmatrix}. \tag{2.3}$$

To guarantee that the room temperature stays within predetermined bounds, the following assumption is considered:

$$A_f^{\mathrm{phyx}} = \begin{cases} 0 & : T_{r,A} > T_r^{\max} \\ Q_{c,1} & : T_r^{\min} \leq T_{r,1} \leq T_r^{\max} \\ \dot{Q}_c^{\max} & : T_{r,A} < T_r^{\min} \end{cases}, \tag{2.4}$$

where u_i^{phys} is the actual power consumption [46]. To solve Eqs. (2.3) and (2.4), it's necessary to have a precise estimation of the hidden state (T_m) as well as accurate measurements of external factors like G and I_g. However, obtaining precise estimates and measurements in practice is challenging, resulting in approximate solutions. Additionally, building parameters such as the conductivity of various walls can change over time due to degradation, introducing bias into the model [46]. Consequently, directly modeling a household using these equations is a complex and costly endeavor that can result in biased and suboptimal control strategies.

2.2 Mathematical Framework For Modeling Decision Making

A frequently employed framework for representing problems involving sequential decision-making is Markov Decision Process (MDP). This methodology encapsulates decision-making scenarios wherein the outcome of a decision is influenced by both the current state and the action taken, with the subsequent state being probabilistically determined. MDPs find extensive application across various domains, including robotics, finance, healthcare, and artificial intelligence, due to their ability to effectively model uncertain and dynamic environments.

MDPs offer a clear framework for addressing the challenge of learning through interaction to achieve a specific objective. Within this framework, the entity responsible for learning and making decisions is referred to as the agent, while everything external to the agent, encompassing the surroundings, is termed the environment. These components engage in ongoing interaction, Figure 2.2, with the agent making choices and the environment reacting to these choices, thereby presenting the agent with new circumstances. Additionally, the environment provides rewards, which are numerical values that the agent endeavors to maximize over time by selecting appropriate actions.

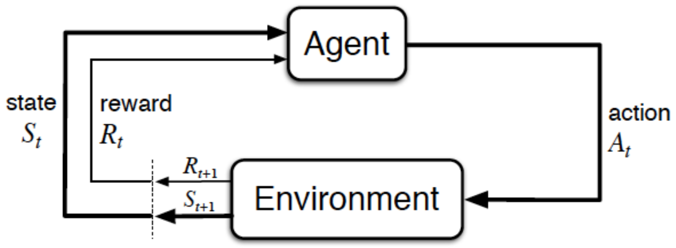


Figure 2.2: The agent–environment interaction.

More precisely, the interaction between the agent and the environment occurs at discrete time steps, denoted by $t = 0, 1, 2, 3$, and so forth. At each time step t, the agent is provided with some representation of the environment's state, denoted as x_t, based on which it selects an action, denoted as A. Subsequently, after one time step, partly due to its chosen action, the agent receives a numerical reward, denoted as R, and transitions to a new state, denoted as x_{t+1}. In a fully observable environment, this interaction can be described as:

$$x_{t+1} = \mathrm{f}(x_t, A_t, w_t), \tag{2.5}$$

where w_t represents the stochasticity in the system and is considered as an independent random variable. Moreover, f is a state transition function which shows a mapping between current and new states. Data-driven approaches simplify the task into a supervised learning scenario with a set of training data, including $\{(x_1, A_1, w_1, x_2), \ldots, (x_N, A_N, w_N, x_{N+1})\}$.

With Stochastic Gradient Descent (SGD), the cost function is defined as follows:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_{i+1} - f_\theta (\mathbf{x}_i, \mathbf{A}_i, \mathbf{w}_i))^2 \tag{2.6}$$

It's worth noting that Eq. 2.6 describes a scenario wherein a system is fully observable, meaning complete state information is available for predicting subsequent states. However, real-world systems, such as thermal models for buildings, are typically only partially observable, where certain state parameters are inaccessible or cannot be directly measured. In such instances, direct utilization of Eq. 2.6 proves impractical due to the incomplete information from observed states. To address this challenge, [12] proposes a strategy where observed variables in a building's thermal model could include room temperature or actual power consumption, while a hidden parameter could represent the temperature of the building's thermal mass, which is difficult to directly measure or estimate. In order to compensate for the absence of this hidden parameter, the proposal suggests using a series of past room temperature measurements instead of a single observation. This engineered feature aims to mitigate the issue of partial observability in the system.

For partially observable MDPs, the state space (X) is comprised of two parts: the observable component (X^{obs}) and a feature-engineered component (X^{f}), forming $X = X^{obs} \times X^{f}$ [40]. With this assumption, the next observable state (x_i^{obs}) is computed as

$$\begin{aligned} \widehat{\boldsymbol{x}}_{i+1}^{\text{obs}} &= f_\theta\left(\mathbf{x}_i, A_i, \mathbf{w}_i\right), \\ \min_\theta \frac{1}{N} &\sum_{i=1}^{N}\left(\boldsymbol{x}_{i+1}^{\text{obs}} - f_\theta\left(\mathbf{x}_i, A_i, \mathbf{w}_i\right)\right)^2 \end{aligned} \tag{2.7}$$

The behavior of the system can be estimated by employing either ordinary or partial differential equations. Thus, the general differential equation for the system can be written as

$$D_\Omega\left(\mathbf{x}_i^{obs}, \mathbf{A}_i, \mathbf{z}_i, \mathbf{z}_{i+1}, \mathbf{w}_i\right) = \mathbf{0}, \tag{2.8}$$

where z_i and D_ω are the hidden state parameters and a generic differential operator in the physical section, respectively. In the following section, the process of integrating an MDP with a building's thermal model to develop physics-informed machine learning is shown.

Chapter 3

Physics Informed Neural Network

In contrast to traditional Neural Networks (NNs), the principles of thermal dynamics are incorporated into a fully connected neural network structure, resulting in Physics-Informed Neural Network (PINN), Figure 3.1. These PINNs can be trained using real-world data.

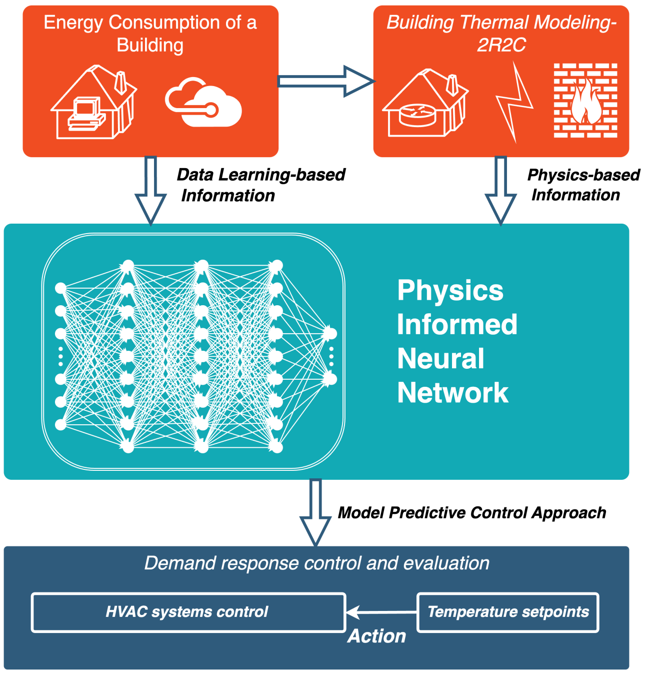

Energy Consumption of a Building
Building Thermal Modeling-2R2C
Data Learning-based Information
Physics-based Information
Physics Informed Neural Network
Model Predictive Control Approach
Demand response control and evaluation
HVAC systems control
Action
Temperature setpoints

Figure 3.1: Workflow of PINN.

This network is designed to forecast future observations, such as zone temperature and thermal load demand (x_{i+1}^{obs}), as well as the temperature of lumped thermal mass (z_i), based on current observations, which include current and past zone temperatures (x_i), external weather data (w_i), actions of the energy system (A_i), and system parameters (Ω). D_Ω denotes a general differential operator in the physical domain. To integrate physics-based knowledge into a conventional neural network, the loss function is typically reformulated as shown in Eq.3.1 [47].

$$\begin{aligned}
\text{LOSS } &= \mathcal{L}_{NN} + \lambda \cdot \mathcal{L}_{\text{phys}} \\
\mathcal{L}_{NN} &= \frac{1}{N} \sum_{i=1}^{N} \left(\boldsymbol{x}_{i+1}^{\text{obs}} - \widehat{\boldsymbol{x}}_{i+1}^{obs} \right)^2 \\
\mathcal{L}_{\text{phys}} &= \frac{1}{N} \sum_{i=1}^{N} \left(D_\Omega \left(\mathbf{x}_i^{\text{obs}}, \mathbf{A}_i, \mathbf{z}_i, \mathbf{z}_{i+1}, \mathbf{w}_i \right) \right)^2
\end{aligned} \tag{3.1}$$

Here, L_{nn} signifies the prediction discrepancies of conventional neural networks, while L_{phys} indicates the physics loss, which connects the network to physical laws and existing knowledge. λ is a regularization term that determines the level of physical knowledge usage. In the conventional neural network, λ is set to zero.

Based on formulation mentioned above, two different models of physics informed neural networks were proposed by [46].

3.1 Model 1: PhysNet

Figure 3.2 depicts a structure consisting of two modules, Encoder and Dynamics modules. The Encoder module is defined by the parameters θ_L, while the Dynamics module is characterized by θ_d [46].

The high dimensional, feature-engineered state input component (x_i^f) is encoded into a low dimensional latent representation (z_i) by the encoder module, which generates a bottleneck. The network's dynamics module then uses this latent representation, action (A_i), and other exogenous information (w_i) in conjunction with observable state information (x_i^{obs}) to forecast the system's next observable state (x_{i+1}^{obs}) [46]. Therefore, the forward pass of this network can be written as follows:

$$\begin{aligned}
z_i &= g_{\theta_L}\left(\boldsymbol{x}_i^f\right), \\
\widehat{\boldsymbol{x}}_{i+1}^{\text{obs}} &= h_{\theta_L}\left(z_i, \boldsymbol{x}_{i+1}^{\text{obs}}, A_i, \mathbf{w}_i\right),
\end{aligned} \tag{3.2}$$

As mentioned in [46], with this design, the prediction of the latent representation (z_i) is one of the parameters that determines the next observable state prediction. This guarantees that information about the system dynamics will be included in the encoded representation that is retrieved from this network.

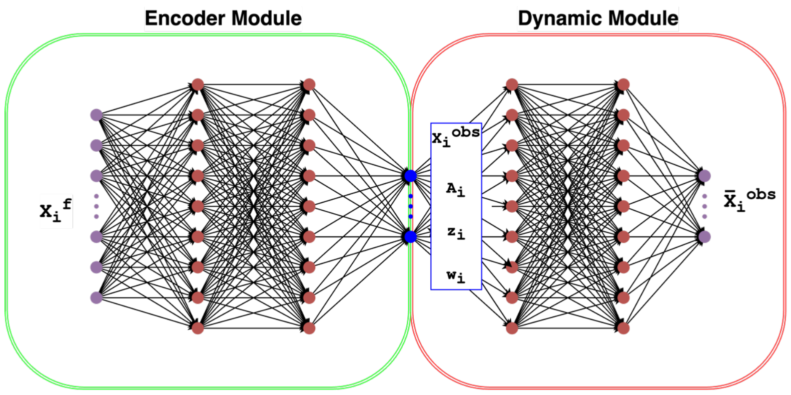


Figure 3.2: PhysNet Structure.

3.2 Model 2: PhysRegMLP

A somewhat normal fully connected neural network topology is applied with the incorporation of physics knowledge based on Eq. 3.2. The complete state representation action, and exogenous data make up this architecture's inputs. The network concurrently predicts a latent representation and the future observable state using these inputs. Due to the identity activation function used by the output layer, the outputs (x_{i+1}^{obs} and z_i) are linear combinations of the output from the network's final hidden layer (s_θ) [46]. Therefore, the forward pass of this network can be expressed as follows:

$$\begin{aligned} z_t &= g_1 s_\theta\left(x_i, A_l, w_l\right) + g_2, \\ \hat{x}_{i+1}^{\text{obs}} &= h_1 s_\theta\left(x_i, A_l, w_l\right) + h_2. \end{aligned} \tag{3.3}$$

where g_1, g_2, h_1 and h_2 are matrices. As clarified by [46] for this model, it is not possible to derive the predictions for the future observable state using the derived latent representation and this parameter sharing. This implies that there's a chance this latent representation doesn't have enough details on the system dynamics. But because of the laws of physics, the latent representation—which depicts the system's hidden parameters—is physically significant. Due to this, the physics module in this instance serves as a regularization term that directs the network to concurrently learn the system's dynamics and a few latent representations.

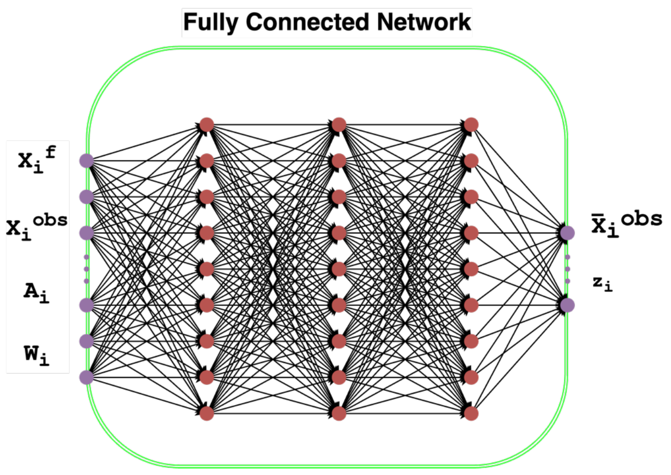


Figure 3.3: PhysRegMLP Structure.

3.3 Configurations of PINN

By applying a first-order Euler discretization technique, the continuous time state-space equations in Eq. 2.3 can be reduced to a discrete-time Eq. 3.4.

$$\begin{aligned}\begin{bmatrix} T_{\mathrm{r}j+1} \\ \mathrm{T}_{m,1+1} \end{bmatrix} &= \mathbf{a} \times \begin{bmatrix} T_{\mathrm{r}} \\ T_m \end{bmatrix} + \mathbf{b} \times Q_e + \mathbf{c} \times \begin{bmatrix} T_{\mathrm{a}} \\ G \\ I_{\mathrm{g}} \end{bmatrix} \\ &= \begin{bmatrix} -a_{11} & a_{12} \\ a_{21} & -a_{22} \end{bmatrix} \begin{bmatrix} T_r \\ T_m \end{bmatrix} + \begin{bmatrix} b_1 \\ 0 \end{bmatrix} \dot{Q}_c + \begin{bmatrix} c_{11} & c_{12} & c_{12} \\ 0 & c_{22} & c_{22} \end{bmatrix} \begin{bmatrix} T_{\mathrm{A}} \\ G \\ I_{\mathrm{g}} \end{bmatrix}. \end{aligned} \tag{3.4}$$

where parameters a, b, and c stand for the building's physical knowledge, which can be estimated using a pure data-driven model or computed using actual building measurements. These parameters can be set to a specific value for the estimation case and adjusted using on-site data during the training phase. As a result, this model may function even in the absence of thermodynamics parameters. The loss function can be expressed as follows using this 2R2C model.

$$\begin{aligned} \mathcal{L}_{NN} &= \frac{1}{N} \sum_{i=1}^{N} \left(T_{r,i} - \hat{T}_{r,1} \right)^2 + \frac{1}{N} \sum_{i=1}^{N} \left(A_{r,i}^{\text{phys}} - \hat{A}_{r,i}^{\text{phys}} \right)^2 \\ \mathcal{L}_{\text{phys}} &= \frac{1}{N} \sum_{t=1}^{N} \left(T_{m,i} - \hat{T}_{m,t} \right)^2 . \end{aligned} \tag{3.5}$$

Using a 2R2C structure, the thermal mass temperature (T_m) can be computed as

$$\begin{aligned} T_{m,i} &= \frac{1}{a_{12}}\left(T_{r,i} + a_{11}\hat{T}_{r,1} - b_1\dot{Q}_i - c_{11}G_t - c_{12}I_t - c_{13}T_{a,1}\right) \\ T_{m,i} &= \frac{\tau_{r,+1} - \tau_{r,i}.}{\Delta t}. \end{aligned} \tag{3.6}$$

where $T_{m,i}$, the lumped thermal mass temperature of the state i, can be computed using Eq. 3.6, and $\hat{T}_{r,i}$ is the room temperature of the state i. The loss function in Eq. 3.6 is therefore constrained by physical laws.

3.4 Neural network Architecture

Almost all machine learning algorithms follow the same framework for training their models, Figure 3.4. As can be seen in this figure, this process is comprised of the following parts:

- **Data preprocessing**: Data preprocessing is a crucial phase in the machine learning process since the quality of the data and the information that can be extracted from it directly influence how well our model can learn. First, the rows that contain null values were dropped. Second, outliers are eliminated by the use of box and whisker plots. The box and whisker plot method, a valuable statistical tool, offers a visual representation of a dataset's distribution, while also effectively identifying outliers. Central to this method is interquartile range (IQR), depicted by the box, which encapsulates the middle 50% of the data. Mathematically, IQR is calculated as the difference between Third quartile (Q_3) and First quartile (Q_1), i.e.,

  $$IQR = Q_3 - Q_1 \tag{3.7}$$

  The whiskers extend from the edges of the box to the minimum and maximum values within a range typically set at 1.5 times IQR. Formally, the upper whisker limit is defined as

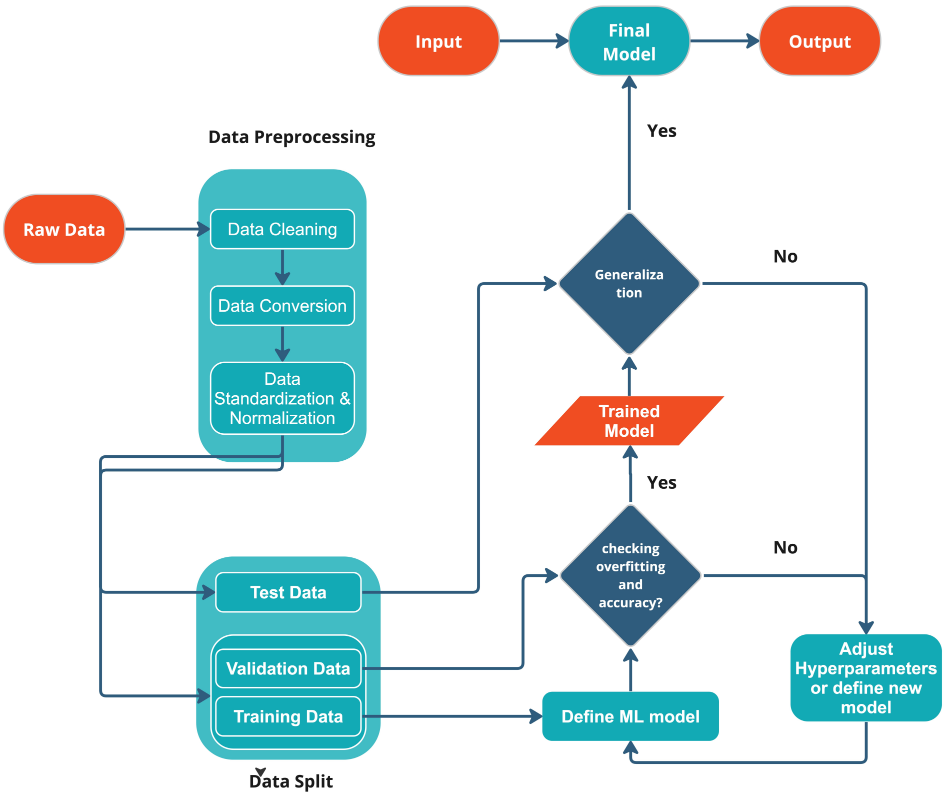
  $$Q_{upper} = Q_3 + 1.5 \times IQR \tag{3.8}$$

  while the lower whisker limit is

  $$Q_{lower} = Q_3 - 1.5 \times IQR \tag{3.9}$$

  Any data points lying beyond these limits are considered potential outliers. These outliers, represented graphically, signify values significantly deviating from most of the dataset, indicating possible anomalies or measurement errors.
  As mentioned in the previous section, there are two kinds of data, including data-based learning and physics-based information, which have been cleaned before applying neural network method.

- **Split cleaned data**: Usually, data splitting is done to prevent overfitting, in which a machine learning model fits the training data too well and is unable to consistently fit new data. Given this situation, the data is divided into two parts, training and test data. Next, a subset of the training data is used as the validation data to compare the performance of several trained models. This helps us to select the appropriate model class or hyper-parameters. For our problem, we separate 24 data from the end of the series for the test and 10 percent from the end of the training data for validation.



image

Figure 3.4: Neural Network Training Flowchart.

- **Train model**: In order to forecast the hourly energy usage, PINN is used.
- **Evaluate model**: The following performance metrics are used to assess each model's effectiveness: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE). Mean Absolute Error (MAE) refers to the magnitude of difference between the prediction of an observation and the true value of that observation.

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right). \tag{3.10}$$

The squared difference between the estimated and actual values is measured by Mean Squared Error (MSE).

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{n}(y_1 - \hat{y}_1)^2 \tag{3.11}$$

Another measure used to determine the differences between the estimated value and the actual value of the model is called Root Mean Squared Error (RMSE). Root Mean Squared Error (RMSE) is defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_1 - y_1)^2} \tag{3.12}$$

Chapter 4

Other Machine Learning Methods

4.1 SARIMA

Seasonal Autoregressive Integrated Moving Average (SARIMA) model is frequently used to predict seasonal time series. A Seasonal ARIMA (SARIMA) model assumes multiplicative seasonality. In other words, this method combines an ARIMA model with a seasonality factor. The equation is written as $ARIMA(p,d,q)(P,D,Q)m$, where the first parenthesis shows ARIMA method's parameters and the second one shows SARIMA method's seasonality factor, with m indicating the number of time steps in a seasonal cycle. Figure 4.1 depicts the process of forecasting energy consumption by using SARIMA model.

These steps are explained in the following.

1. **Visualize data**: Data visualization is required before applying machine learning methods to it, as it reveals patterns in the data, such as seasonality and trends.
2. **Augmented Dickey Fuller Test**: SARIMA model can only be used for stationary time series. There are several methods used for this purpose. In this thesis, Augmented Dickey-Fuller (ADF) was selected to apply to the data. ADF test is an advanced model test in which the null hypothesis is a unit root in an autoregressive model. The presence of unit roots in time series implies that unexpected results might be uncovered during time series analysis. It means that forecasting would be inaccurate. ADF can evaluate stationary properties as well as handle more complex statistics.
3. **Stationarize the time series data**: If the series is non-stationary, it can be converted into a stationary signal. Detrending, seasonality, and differencing are three methods for rationalizing data sets. In general, the differencing method, as used in this paper, is applied to data transformation and stationarization. If a signal S_t is non-stationary, it can be stationarized into a stationary signal T_t using the following equation

$$T_t = S_t - S_{t-1}, \tag{4.1}$$

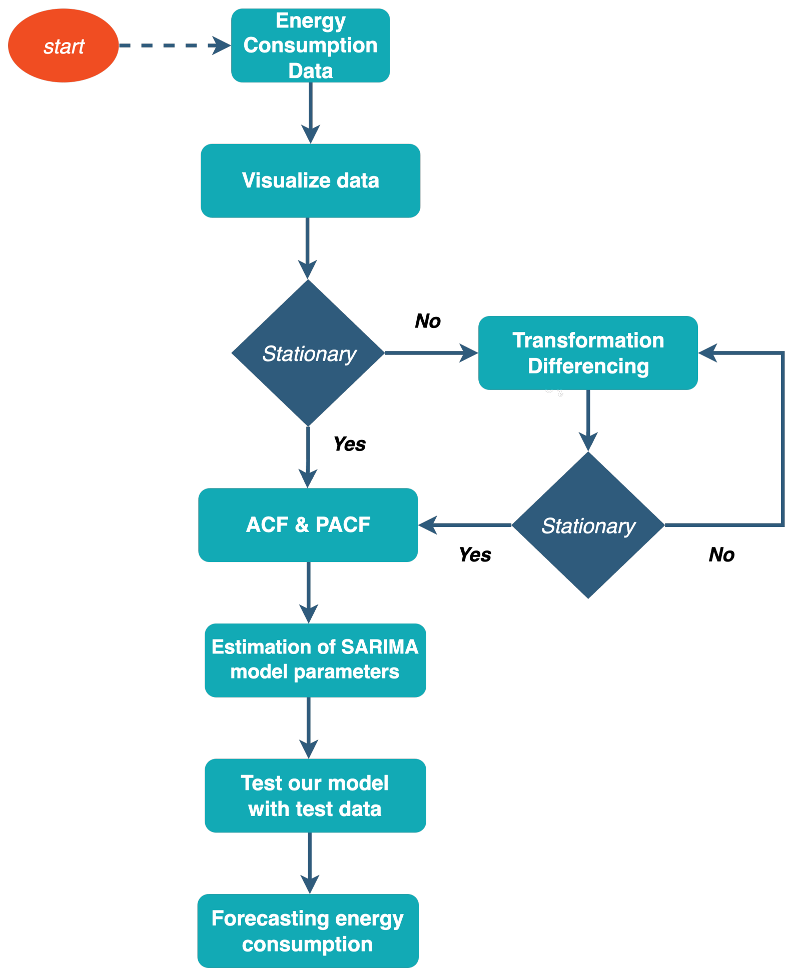

start
Energy Consumption Data
Visualize data
Stationary
No
Transformation Differencing
Yes
ACF & PACF
Stationary
Yes
No
Estimation of SARIMA model parameters
Test our model with test data
Forecasting energy consumption

Figure 4.1: Flowchart of SARIMA method.

The signal T_t will not always be stationary despite these transformations. Although it is infrequent, it can happen. In that case, if T_t is non-stationary, the same transformation can be applied to the signal T_t.

4. **ACF/PACF**: Two graphs, ACF and PACF, can be used to make predictions about the ARIMA parameters p, d, and q. Auto-correlation function (ACF) provides the auto-correlation value for any lagging series of variables. This means that it describes how well current values connect to previous values. One of the ways to predict ARIMA parameters, p, d, and q, is the use of two graphs: ACF and PACF. ACF is the comprehensive auto-correlation function, which gives us the value of the autocorrelation of any series with lagged values. This means it describes how well present values relate to past values. These values are plotted along with a confidence band to create an ACF plot. Each time series has several components, including seasonality, trend, cyclicality, and residuality. In order to find correlations, ACF takes into account

all of these factors, which is why it is called the complete auto-correlation plot. Partial auto-correlation function (PACF) is a statistical measure that captures the correlation between two variables after controlling for the effects of other variables. For example, if we are regressing a signal S at lag t (S_t) with the same signal at lags $t-1$, $t-2$ and $t-3$ ($t-1, S_{t-2}, S_{t-3}$), the partial correlation between S_t and S_{t-3} is the amount of correlation between S_t and S_{t-3} that is not explained by their mutual correlations with S_{t-1} and S_{t-2}.

$$S_t = \phi_1 S_{t-1} + \phi_2 S_{t-2} + \phi_3 S_{t-3} + \epsilon \tag{4.2}$$

where ϕ_1, ϕ_2, and ϕ_3 are coefficients and ϵ is the error. From the regression formula above, PACF value between S_t and S_{t-3} is the coefficient ϕ_3. This coefficient will give us direct effect of time-series S_{t-3} to the time series S_t because the effects of S_{t-2} and S_{t-1} are already captured by ϕ_1, and ϕ_2.
Estimate parameters for SARIMA model. ARIMA is a regression type equation in which the independent variables are lags of the dependent variable or lags of the forecast errors. The equation for ARIMA is given as follows [48]:

$$y^{'}(t) = c + \phi_1.y^{'}_{t-1} \cdots + \phi.y^{'}_{t-p} + \theta_1.\epsilon_{t-1} \cdots + \theta_q.\epsilon_{t-q} + \epsilon_t, \tag{4.3}$$

the equation contains three main terms:
Auto Regression (AR): This component involves regressing the time series against its previous values, shown as y_{t-1}, y_{t-2}, etc. The order of this lag is represented by p.
Integration (I): This component utilizes differencing to render the time series stationary. The order of differencing is represented by d.
Moving Average (MA): This component involves regressing the time series against residuals of past observations, represented as error ϵ_{t-1}, ϵ_{t-2}, etc. The order of this error lag is represented by q.
In Eq. 4.3, $y^{'}(t)$ signifies the differenced series, ϕ_1 is the coefficient of the first AR term, p is the order of the AR term, θ_1 is the coefficient of the first MA term, q is the order of the MA term and ϵ_t is the error. ARIMA does not support seasonal data. A seasonal ARIMA model is used when a time series has a significant seasonal pattern. In addition to the three parameters mentioned in the previous section, p, d, q, SARIMA has four more seasonal parameters $(P, D, Q)m$. The first three parameters account for Autoregressive Component (P), Difference Component (D), and Moving Average Component (Q) at the seasonal level, and m is number of observations per season which for our case is 12. The generalized form of SARIMA model can be written as [49]:

$$\begin{aligned}
&\phi_p(B)-\Phi_p\left(B^5\right)\left(1-B^5\right)^D\cdot z_t=\theta_q(B)\cdot\theta_Q\left(B^5\right)\cdot a_t\\
&\phi_p(B)=1-\phi_1B-\phi_2B^2-\cdots-\phi_pB^p\\
&\Phi_p(B)=1-\Phi_1B-\Phi_2B^2-\cdots-\Phi_pB\\
&\theta_q(B)=1-\theta_1B-\theta_2B^2-\cdots-\theta_pB^9\\
&\theta_Q(B)=1-\theta_1B-\theta_2B^{25}-\cdots-\theta_pB^{0s}
\end{aligned} \tag{4.4}$$

where B is the backward shift operator which is used as follows

$$B.y_t=y_{t-1}. \tag{4.5}$$

In other words, B, operating on y_t, has the effect of shifting the data back one period. Two applications of B to y_t shifts the data back two periods.

$$B\left(B.y_t\right)=B^2.y_t=y_{t-2} \tag{4.6}$$

ARIMA models are employed with various parameter configurations for predicting electricity data. The model suggests that seasonal patterns can be viewed as an ARIMA process, with m indicating the number of time steps per seasonal cycle. The model recognised that neighboring time points can influence each other, either within the same season or across different seasons, through typical temporal proximity. Identifying a SARIMA model is particularly challenging compared to an ARIMA model because it requires addressing seasonal effects. Thankfully, the `auto.arima()` function in the forecasts package can manage this complexity, treating it similarly to a standard ARIMA estimation task. As discussed earlier, there are good reasons to go with automated parameter selection unless you have strong knowledge that suggests you override the selected model determined by automated methods.

4.2 Prophet Forecasting Model (PFM)

4.2.1 Architecture of PFM method

The workflow diagram of the model is shown in Figure 4.2.

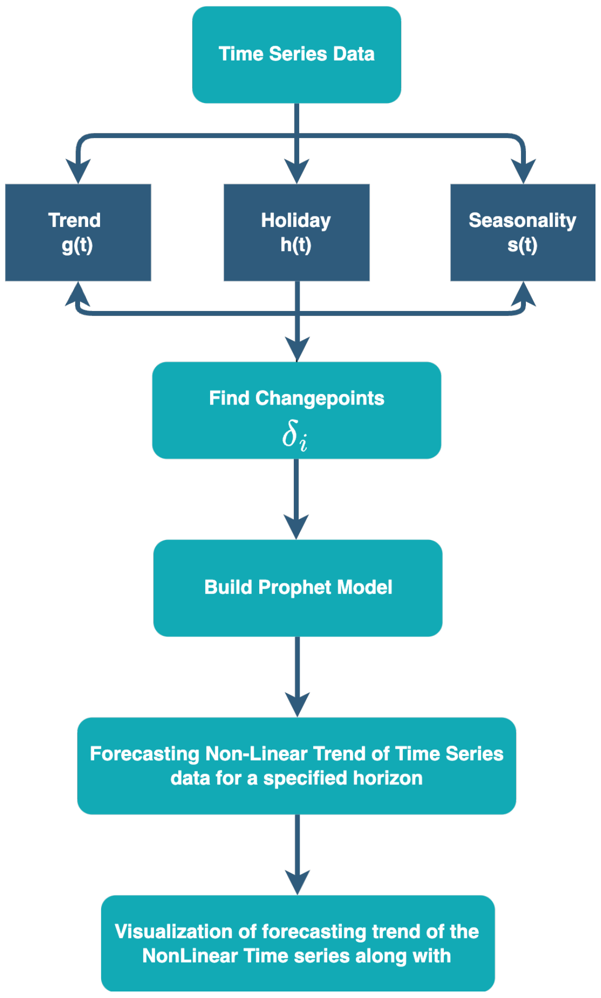

Time Series Data
Trend g(t)
Holiday h(t)
Seasonality s(t)
Find Changepoints δ_i
Build Prophet Model
Forecasting Non-Linear Trend of Time Series data for a specified horizon
Visualization of forecasting trend of the NonLinear Time series along with

Figure 4.2: Flowchart of PFM.

In this model, a time series is decompose in three different components, including trend, seasonality, and holiday. They are combined in the following equation [50]:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t. \tag{4.7}$$

Here $g(t)$ shows the overall trend of data is the trend function without any periodic changes, $s(t)$ represents periodic changes (e.g., weekly and yearly seasonality), and $h(t)$ represents the effects of holidays which is applicable when there is irregular pattern over one or more days.

4.2.2 Trend Model

Two trend models are applied to cover many Facebook applications: line growth model, and linear growth model. For the first model, the growth term will look like a line equation, $y = mx + b$, except the slope (m) and offset (b) are variables. The basic form of logistic model is defined as

$$g(t) = \frac{C(t)}{1 + e^{-k(t-m)}}. \tag{4.8}$$

with C the carrying capacity, k the growth rate, and m an offset parameter.

4.2.3 Seasonality

For modeling seasonality, Fourier series is utilized because it gives us a flexible model for modeling periodic effects. The standard Fourier series is shown as follows

$$s(t) = \sum_{n=1}^{N} \left(a_n \cdot \cos\left(\frac{2\pi nt}{p}\right) + b_n \cdot \sin\left(\frac{2\pi nt}{p}\right)\right) \tag{4.9}$$

To apply seasonality, a matrix of seasonality vectors must be created. For example with yearly seasonality and N = 10,

$$X(t) = \left[\cos\left(\frac{2\pi(1)t}{365.25}\right), \ldots, \sin\left(\frac{2\pi(10)t}{365.25}\right)\right]. \tag{4.10}$$

The seasonal component is then

$$s(t) = X(t)\beta. \tag{4.11}$$

where $\beta = [a_1, b_1, ..., a_N, b_N]$ and taken as $\beta \approx Normal(0, \sigma)$ [50].

4.2.4 Holidays and Events

Occasionally, some predictable shocks might be happened in the time series. This effect can be modeled by holidays and events function. This function allows Facebook Prophet model to adjust forecasting on those special days. We can identify the set of past and future dates, D_i, for the holidays.

4.2.5 LSTM

LSTM is a type of Recurrent Neural Network (RNN) architecture specifically designed to address the vanishing gradient problem that occurs in traditional RNNs. It is well-suited for modeling and forecasting time series data due to its ability to capture long-term dependencies and retain memory over extended sequences.

4.2.6 Architecture of an LSTM Unit

At the core of an Long Short-Term Memory (LSTM) network lies the LSTM unit, which consists of several components, 4.3:

- **Memory Cell (C_t):** The key component of LSTM is the memory cell, denoted by C_t , which stores information at time step t. The cell has three main components: the input gate, forget gate, and output gate.
- **Input Gate (i_t):** The input gate determines the extent to which new information should be stored in the memory cell. It is calculated as follows.

$$i_t = \sigma(W_i \cdot [h_(t-1), x_t] + b_i) \tag{4.12}$$

  where W_i and b_i are the weight matrix and bias vector for the input gate, h_{t-1} is the previous hidden state, x_t is the input at time t, and σ is the $sigmoid$ activation function.
- **Input Gate (i_t):** The input gate determines the extent to which new information should be stored in the memory cell. It is calculated as follows.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{4.13}$$

  where W_i and b_i are the weight matrix and bias vector for the input gate, h_{t-1} is the previous hidden state, x_t is the input at time t, and σ is the $sigmoid$ activation function.
- **Forget Gate (f_t):** The forget gate controls the extent to which the previous memory cell content should be retained. It is calculated as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4.14}$$

  where W_f and b_f are the weight matrix and bias vector for the forget gate.
- **Output Gate (o_t):** Determines which information from the memory cell should be output to the next time step. It is calculated as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4.15}$$

  where W_o and b_o are the weight matrix and bias vector for the output gate.
- **Hidden State (h_t):** The hidden state is computed based on the memory cell and the output gate:

$$h_t = o_t \cdot \tanh(c_t) \tag{4.16}$$

  where $tanh$ is the hyperbolic tangent activation function.
- **Candidate Activation ($\tilde{C_A}$):** Calculated based on the input and previous hidden state, this represents the new information that could be stored in the memory cell.

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{4.17}$$

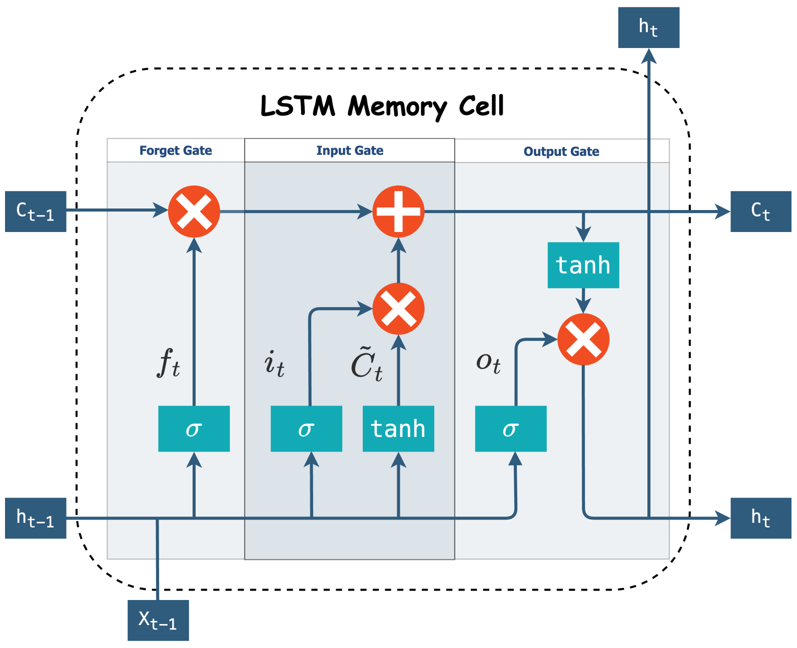


Figure 4.3: LSTM model architecture.

The memory cell is updated using the input gate, forget gate, and new candidate values. The update equation is:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{4.18}$$

In time series forecasting, LSTM models are trained on historical data to learn patterns and relationships between past observations and future outcomes. The trained model can then be used to make predictions for future time steps.

During training, the LSTM model is optimized to minimize the difference between predicted and actual values using techniques such as gradient descent and Backpropagation through time (BPTT).

Chapter 5

Results and discussions

Three other machine learning methods were conducted to evaluate the effectiveness of PINN designs and to gauge their efficacy as models for control purposes.

5.1 Real-world building dataset

Our data includes information gathered from the cold storage of a private building located in the south of Norway. The building used in this study is a private house in which heating is provided by an electric heater system. This dataset consists of observations concerning room temperature, actual energy consumption, and ambient air temperature for about 1 year between 1 January 2016 and 30 December 2016. Measurements were taken every 15 minutes, resulting in enough data to simulate around 364 days for training and validation and 1 day for testing. The physical parameter of the house is listed in Table 5.1.

Table 5.1: Physical Parameters

Physics Parameters	**Value**	**unit**
Thermal Capacitance of the Room (C_r)	1500	J/°C
Thermal Capacitance of the Thermal Mass (C_m)	200	J/°C
The specific heat capacity of air $\left(C_\rho\right)$	1005	J/Kg°C
Thermal Resistance Between the Room and the Ambient (R_{ra})	0.15	m^2 K/W
Thermal Resistance Between the Room and the Thermal Mass (R_{rm})	0.07	m^2 K/W
Note: Internal Heat Gain (I_g) and Solar Irradiance (G) are ignored in this model [46].		

5.2 Cleaning Data

After plotting the indoor temperature data, Figure 5.1, it became apparent that there were some outliers present, potentially affecting the accuracy of our analysis.

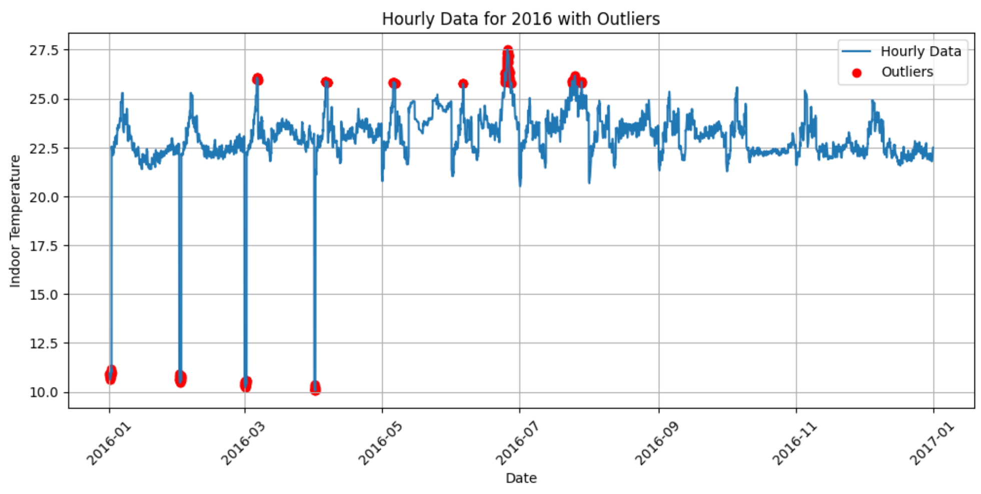


Figure 5.1: Indoor Temperatures with outliers.

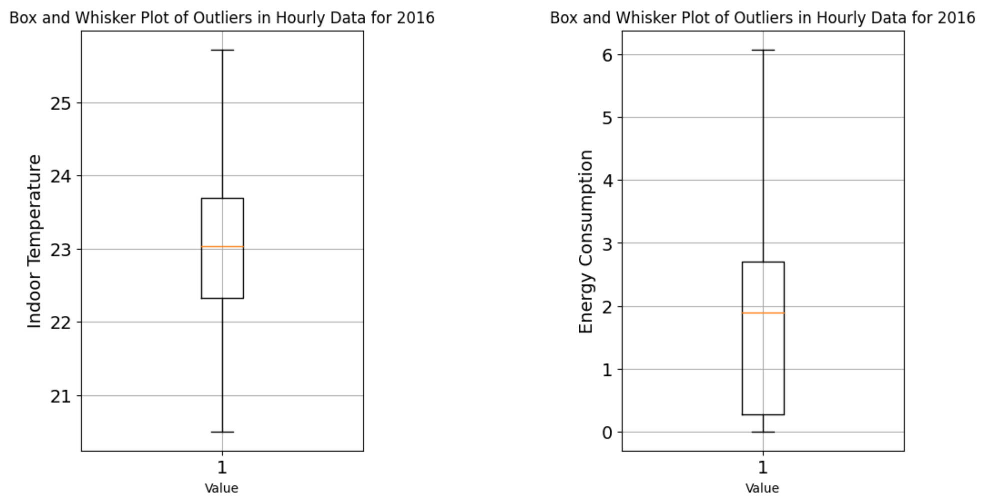
While analyzing indoor temperature data over the duration of a year, it became apparent that certain abnormal temperature readings were not caused by internal building issues. Instead, they were linked to periods when the building was unoccupied. Instead of just deleting these outlier readings, a different approach was taken to maintain the integrity of the data series. However, removing them entirely could disrupt the flow of the data, leading to misleading conclusions.

To address this issue, outliers associated with the unoccupied building were replaced with the mean values from both the previous and next days at the same time. By this, outliers could be fixed while keeping the data in chronological order. Notably, the energy consumption and outdoor temperature data associated with those outliers were also replaced with the mean values from the corresponding previous and next days. The Box and Whisker plot of cleaned internal temperature and energy consumption are illustrated in Figure 5.2. Moreover, the cleaned data is shown in Figure 5.3 and Figure 5.4.

5.3 Physics-Informed Neural Network Result

This section explores how various settings within the models (architectures) will impact their performance. As mentioned in the previous section, both models receive a sequence of past room temperatures and control actions as input.

- **Depth**: This sequence length, called "depth," determines how much past information the model has access to, impacting its ability to estimate the building's hidden state (T_m). Longer sequences (higher depth) can partly compensate for missing information like solar radiation or internal heat gains [46].



(a) Indoor temperature

(b) Energy consumption

Figure 5.2: Box and Whisker plots for Indoor temperature and energy consumption.

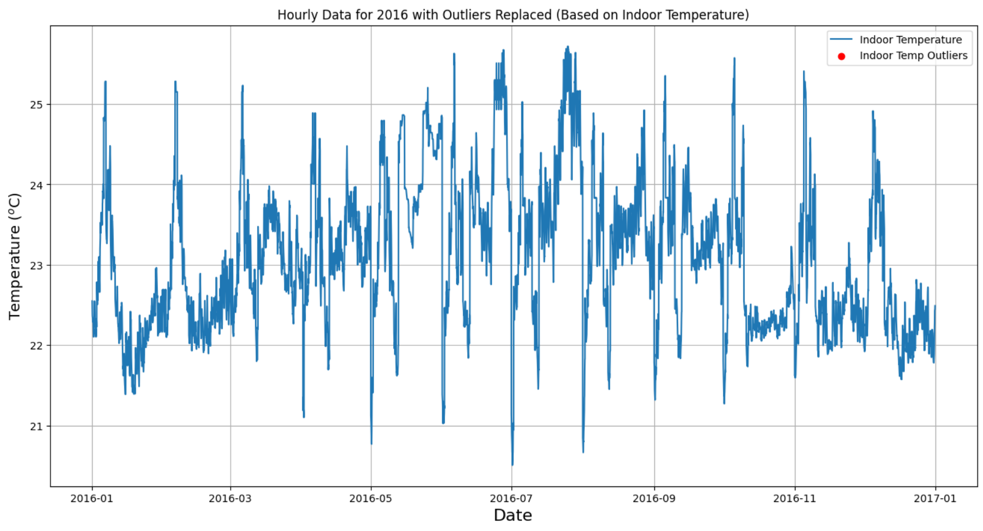


Figure 5.3: Cleaned Indoor Temperature Data.

- **Network size and hyperparameters**: The hyperparameters utilized for both architectures, PhysNet and PhysReg, are displayed in Tables 5.2 and 5.3. It is worth noting that the same hidden layer and neuron numbers have been applied across both architectures, Improving the consistency in comparing the two methods.

The impact of depth on two different physics-informed machine learning methods and Multilayer Perceptron (MLP) method is investigated in Figure 5.5. As the

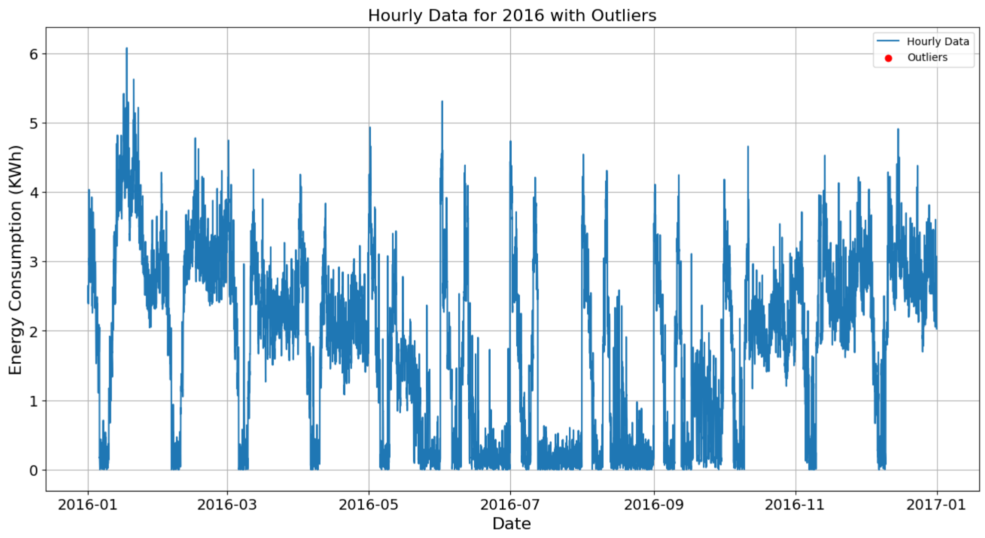


Figure 5.4: Cleaned Energy Consumption Data.

Table 5.2: Hyperparameters for PhysNet architecture

Parameters	**Value**
Optimizer	Adam
Learning Rate	0.001
Activation Function	tanh
Batch Size	1024
Input Neorons	24
Encoder Module (θ_L)	
Hidden Layers	2
Neurons per layer	64
Dynamics Module (θ_d)	
Hidden Layers	2
Neurons per layer	64

Table 5.3: Hyperparameters for PhysReg architecture

Parameters	**Value**
Optimizer	Adam
Learning Rate	0.001
Activation Function	tanh
Batch Size	1024
Input Neorons	24
Hidden Layers	4
Neurons per layer	64

model depth increases, the energy consumption error decreases significantly, as it is supposed, because deeper architectures are capturing more complex patterns in the data. This plot indicates that for depths below 20, `PhysReg` yielded better results compared to `PhysNet`. In simpler term, If these methods is utilized with lower depth, it is preferable to choose `PhysNet` over `PhysReg`. This situation could arise due to various factors; for instance, increasing the number of input neurons expands the input space, which may heighten the likelihood of overfitting. In order to facilitate a clearer comparison of various approaches, a network structure depth is 24 that is based on 24 hours.

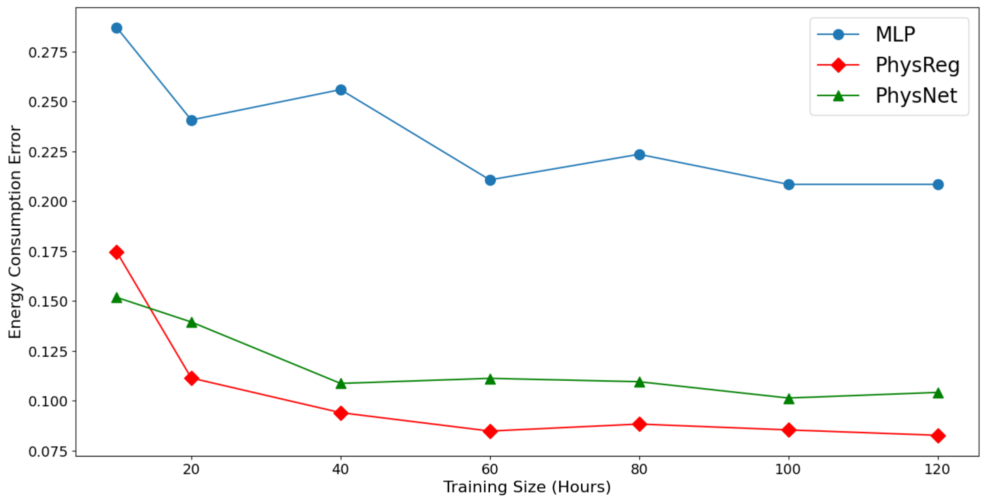


Figure 5.5: The effect of depth on accuracy.

Figures 5.6 and 5.7 represent a comparison between actual and predicted indoor temperature on test data. The fluctuations in actual temperature correspond to the natural variations in indoor temperature due to external factors (e.g., weather, heating, cooling). As can be seen, `PhysNet` and `PhysReg` models are trained to learn the underlying patterns and dynamics of indoor temperature based on historical data. They use a combination of neural network architecture and domain-specific physics knowledge to make accurate predictions. The neural network estimates the thermal mass temperature based on the observed indoor temperature and other relevant features. The green line represents the thermal mass temperature, the hidden state, T_m, calculated by the same neural network. Thermal mass refers to the ability of a material (such as walls, floors, or furniture) to store and release heat. As mentioned in Section 2, it is difficult to calculate this value and this makes our problem partially observable. Thus, it is not possible to calculate the accuracy of this value. However, in some papers, like [46], the building mathematical modeling was solved with finite difference methods to calculate the hidden state for comparison with the output of neural network.

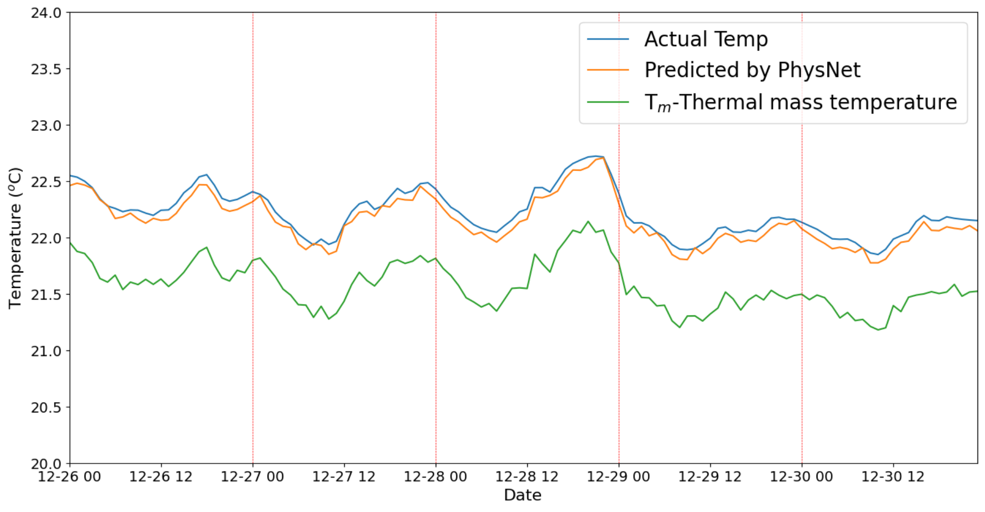


Figure 5.6: Trained physics-informed neural network architectures (PhysNet) when applied to real-world data scenarios.

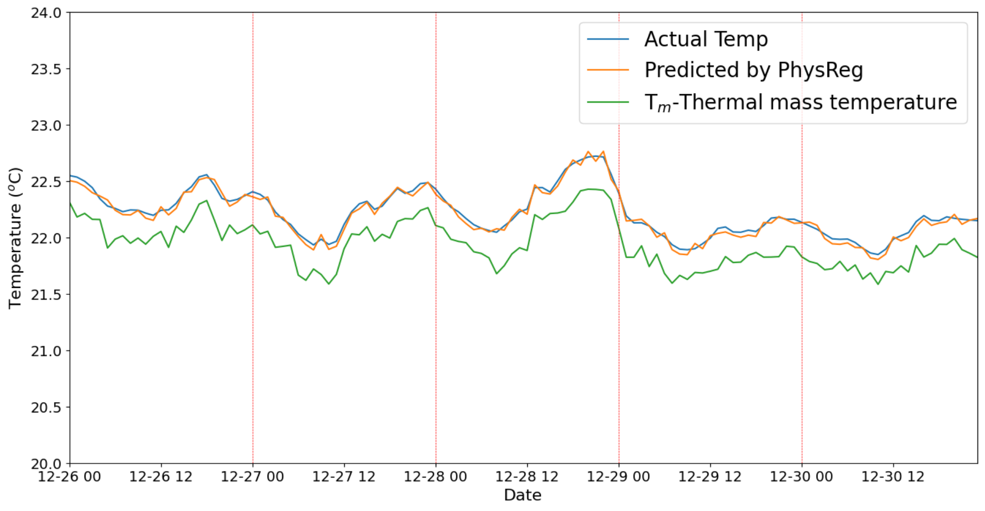


Figure 5.7: Trained physics-informed neural network architectures (PhysReg) when applied to real-world data scenarios.

5.4 SARIMA

5.4.1 Data for prediction

Initially, it's important to determine if `SARIMA` is applicable to this dataset. As depicted in Figure 5.8, the energy consumption patterns remain consistent on a daily basis, highlighting multiple high points in energy consumption around midday and correspondingly lower levels typically seen at the beginning and end of each day. Hence, utilizing `SARIMA` to model the data's behavior seems justified.

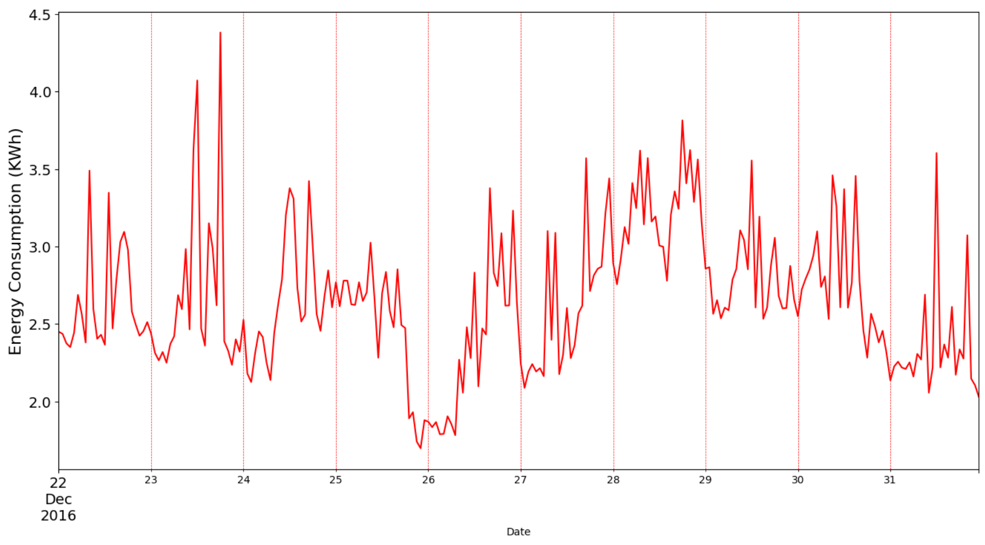


Figure 5.8: The fluctuation of energy consumption in the building in December 2016 ACF and PACF plots.

Figure 5.9 (top) exhibits ACF plot for our dataset. It's evident that the coefficient at lag one stands out significantly compared to others, leading us to select $Q = 1$ for our model. It is worth noting that although coefficients for lags beyond 24 also demonstrate significance, we choose not to incorporate larger lags because they are linked to the dataset's seasonality, which is already addressed by parameter m.

Figure 5.9 (bottom) depicts the PACF plot of our dataset for a lag of 24. Notably, there are robust correlations observed between time periods t and $t-2$ compared to others. Hence, we opt for $P = 2$ in the SARIMA model. To ensure stationarity in our data, we apply first-order differencing, indicating $D = 1$.

The final SARIMA model employed for prediction is SARIMA(0, 0, 0)(2, 1, 1)24, where only the seasonal parameters P, Q, D, and m are utilized, and the non-seasonal component, which corresponds to the ARIMA model, is set to zero. The outcome of our prediction is depicted in Figure 5.10, where it is evident that the forecast closely aligns with the actual values.

The residual of SARIMA model is depicted in Figure 5.11. As depicted in the figure, the greatest error tends to occur around the midpoint of the period. This is primarily due to SARIMA relying solely on historical data to identify patterns for future predictions. Consequently, as evident from Figure 5.10, the energy consumption of the building exhibits a somewhat erratic pattern in the preceding days.

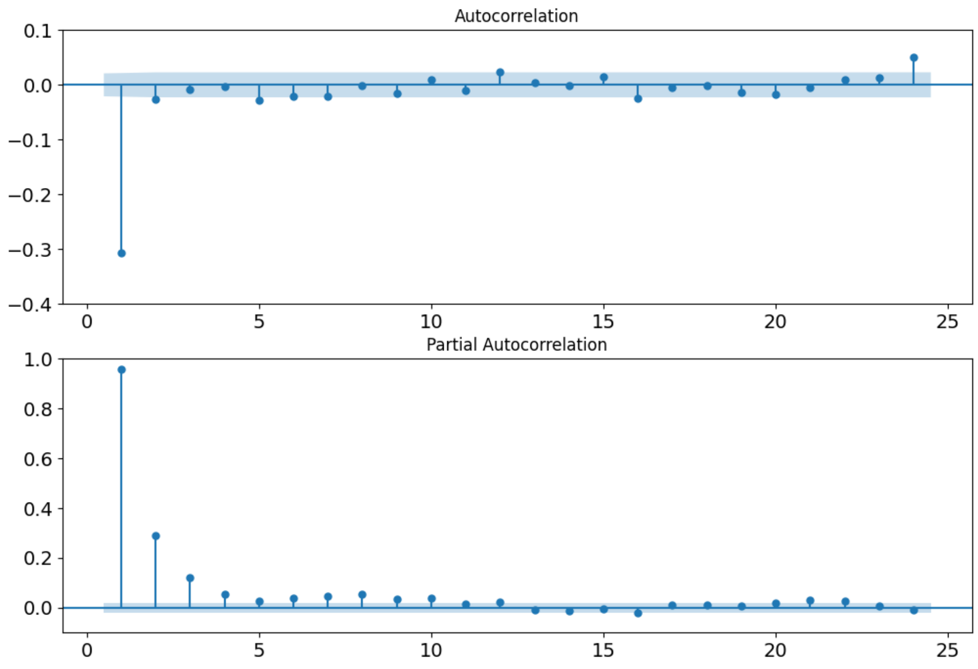


Figure 5.9: ACF (top) and PACF (bottom) plots

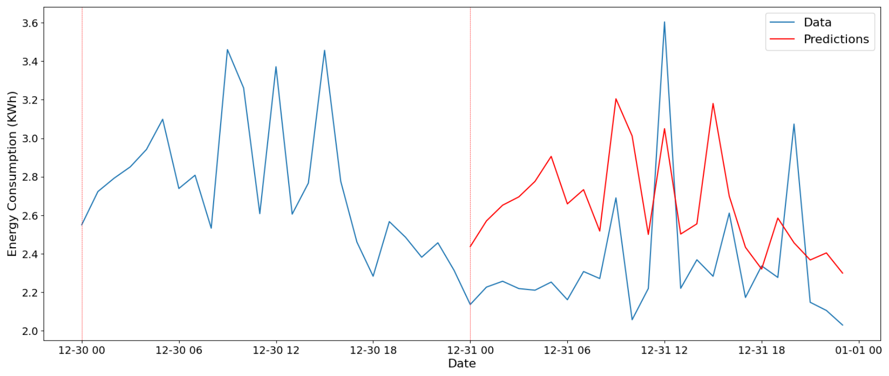


Figure 5.10: SARIMA Prediction

5.5 Prophet Forecasting Model

Figure 5.12 illustrates some important components, including trend, weekly seasonality, and daily seasonality, of the `Prophet` forecasting model applied to the energy consumption of the house. The model's output is visualized in three distinct plots, each providing valuable insights into the energy usage patterns.

The top plot represents the trend of energy consumption over a year, from January 2016 to January 2017. It shows fluctuations in energy use, indicating periods of increased and decreased consumption. This could be due to seasonal changes, variations in daily routines, or other factors affecting energy use.

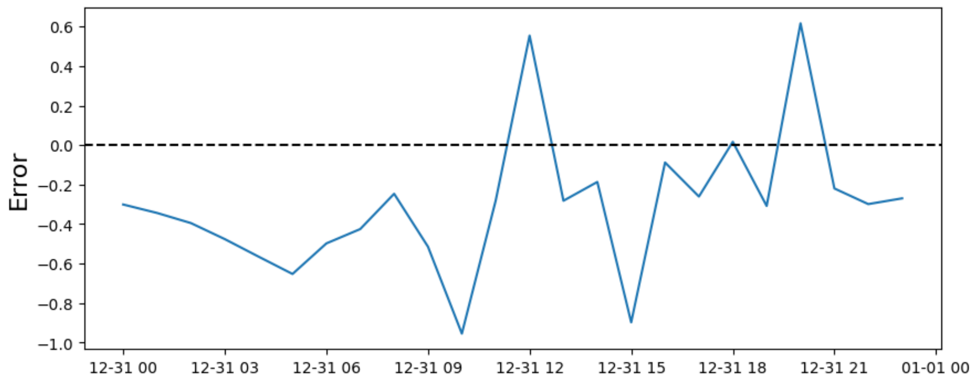


Figure 5.11: Error of SARIMA method (test data)

The middle plot depicts weekly seasonality in energy consumption. It highlights that energy use tends to be lower during weekdays and peaks on Sundays. This pattern might be attributed to the occupants' weekly routines, with more time spent at home during the weekends leading to higher energy use.

The bottom plot shows daily seasonality in energy consumption. It reveals that energy use is highest during the early morning and evening hours. This could be due to increased activity in the house during these times, such as cooking, cleaning, or using electronic devices.

Overall, these plots provide a comprehensive view of the household's energy consumption patterns. Such insights can be instrumental in developing strategies for optimizing energy use and improving efficiency.

The time series plot generated by the `Prophet` forecasting method is plotted in Figure 5.13. It represents the energy consumption (in KWh) over a certain period. The black dots in the plot represent the observed data points, i.e., the actual energy consumption, training data, recorded at different times.

The blue line in the plot represents the forecasted energy consumption. This is the prediction made by the Prophet model based on the patterns of training data.

The shaded area around the blue line represents the uncertainty interval of the forecast. This means that the actual value is expected to fall within this range with a certain level of confidence. The width of the shaded area at any point in time indicates the level of uncertainty in the forecast at that point. A wider shaded area means higher uncertainty.

Figure 5.14 shows a comparison between the actual test data and forecasted energy consumption data. This is often done to evaluate the performance of a forecasting model.

By comparing the actual data points with the forecasted values, we can see how well the `Prophet` model has performed in predicting energy consumption behavior. If the actual data points fall within the uncertainty interval and follow the same general trend as the forecasted values, it indicates that the model has done a good job in capturing the underlying patterns in the data. However,

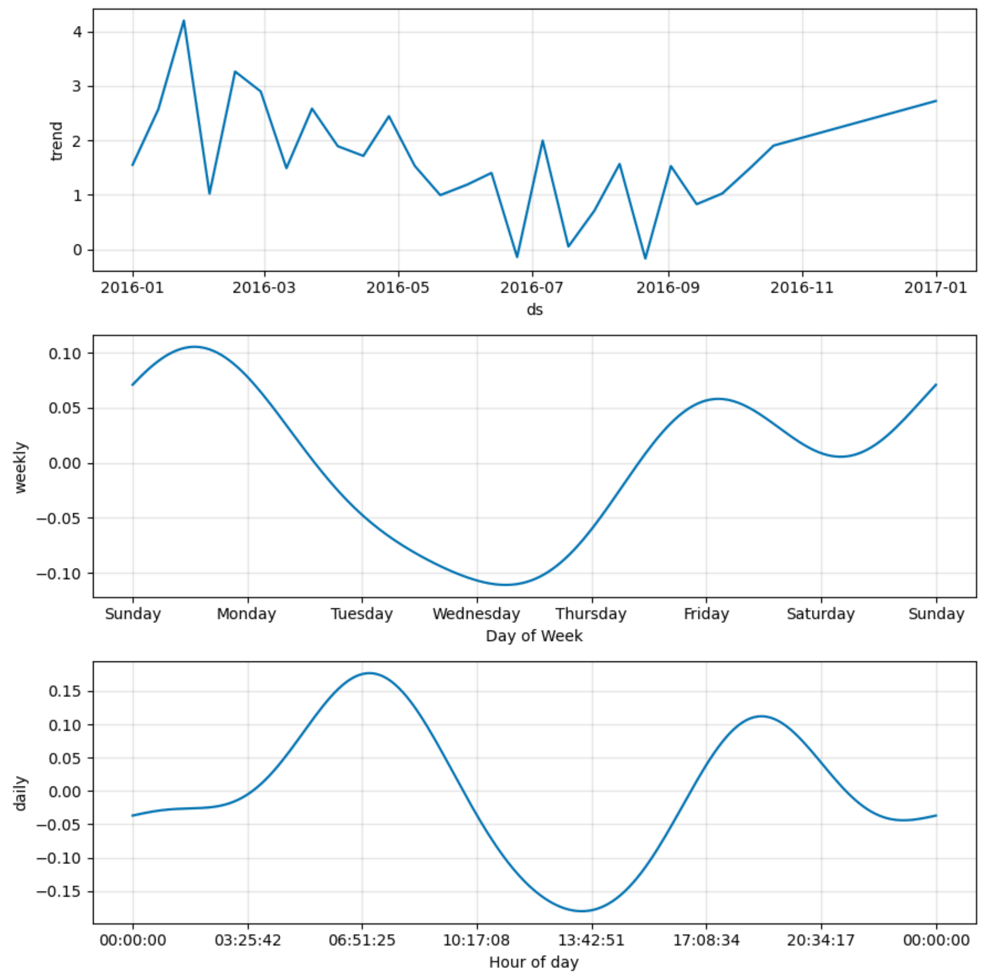


Figure 5.12: Components of Prophet mode.

as demonstrated, `Prophet` might struggle with time series data that have high-frequency fluctuations, such as hourly energy consumption data. This is because these fluctuations can be influenced by many unpredictable factors that are not easily captured by the model. For example, hourly energy consumption can be affected by sudden changes in weather, unexpected appliance usage, or other irregular events, which can be captured by PINN methods because these models do not count only on the historical data, and physics parameters are involved as a part of problem solving. On the other hand, when you aggregate the data into larger time windows, like daily or weekly, many of these short-term fluctuations average out, and the underlying patterns become more apparent and predictable. This is why `Prophet` often performs better on daily or weekly data.

The graph of absolute errors is plotted in Figure 5.15 which represents the absolute errors of the Prophet forecasting method over test data. As mentioned, the highest errors occur in the middle of the period due to a sudden change in the

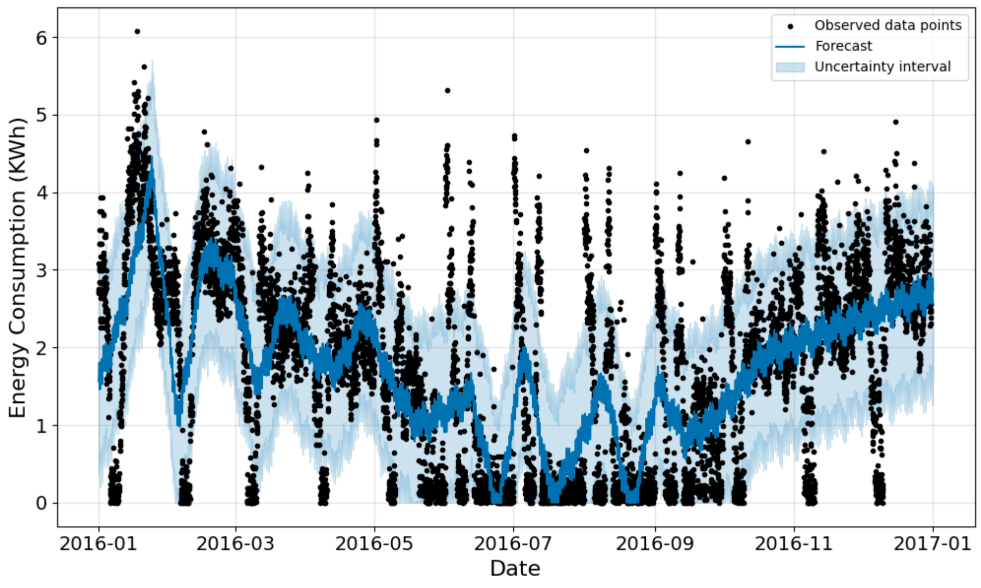


Figure 5.13: Comparison between training data and prophet forecasting method.

trend that the model did not anticipate.

5.6 LSTM

The model, like other time-series models, uses the previous energy data as the input parameter to forecast the next day. Table 5.4 is a summary of hyperparameters used in an LSTM model for energy consumption prediction. The selection of these hyperparameters involved a grid search approach, with Mean Squared Error (MSE) used as the metric for evaluating predictions on a validation set. In the LSTMconfiguration specified, the model comprises two hidden layers, each containing 32 neurons. Within each LSTM layer, there are 64 LSTM units, which are responsible for capturing temporal dependencies in the input sequences.

Figure 5.16 shows a line graph representing the training and validation loss values over epochs in an LSTM model. The training loss (blue line) and the validation loss (orange line) are both plotted against the number of epochs.

The training loss decreases as the model learns from the training data, which is expected behavior. However, the validation loss decreases during the first 10 epochs and then starts to increase again, which shows overfitting. In this case, the model seems to perform well up to a certain number of epochs, after which it starts to overfit. One way to prevent overfitting is to stop training when the validation loss starts to increase, a technique known as early stopping.

Figure 5.17 displays a line graph comparing the actual and predicted energy consumption values over specific dates. The actual values are typically represented by the blue line, while the predicted values are represented by the red line. This

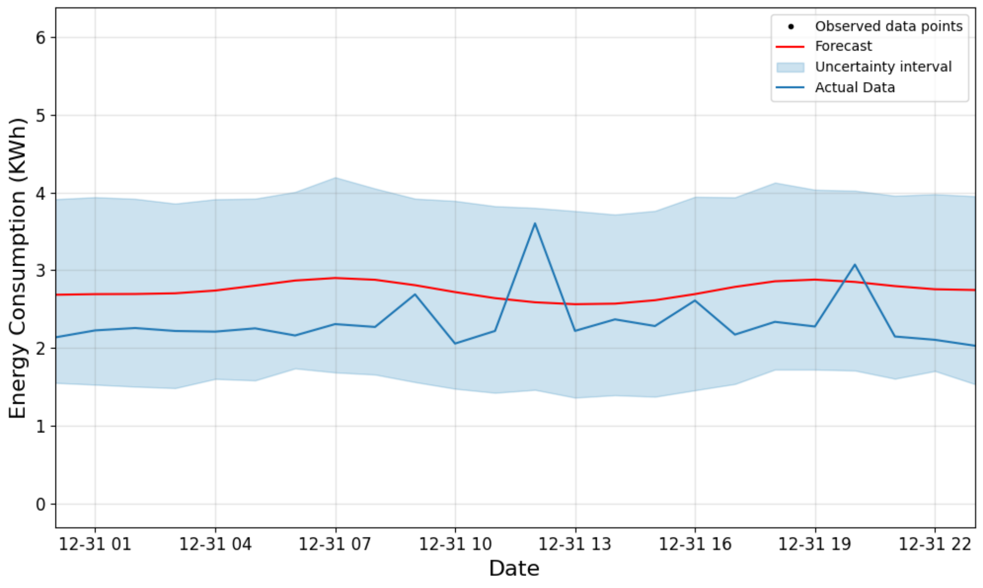


Figure 5.14: Comparison between test data and prophet forecasting method.

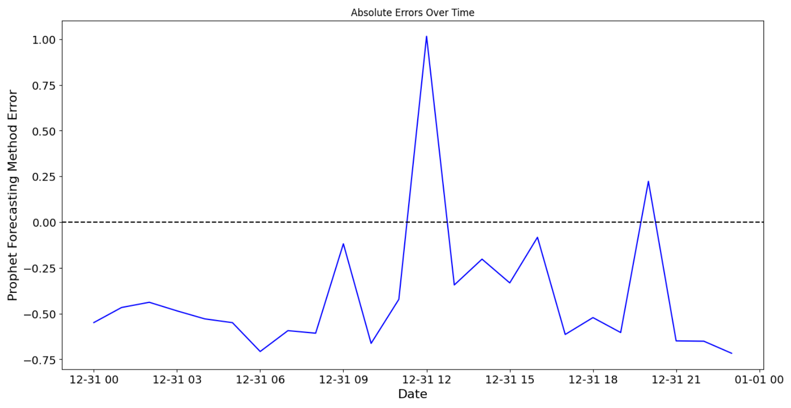


Figure 5.15: Error of Prophet method (test data).

figure provides a visual representation of how well the LSTM model's predictions match the actual energy consumption. The model works by maintaining a form of memory about previous inputs while processing new ones. This allows it to make predictions based on the historical context it has learned. For instance, if energy consumption typically increases at certain times of the day or certain days of the week, as shown in Figure 5.17, the LSTM model can learn this pattern and use it to make accurate predictions. However, The model might not be responsive enough to capture sudden changes in energy consumption.

The fluctuations in the error over time, Figure 5.18, can provide insights into the performance and stability of the method. For instance, large fluctuations can be seen in the middle, which shows the sudden change in energy consumption.

Table 5.4: Hyperparameters for LSTM architecture

Parameters	**Value**
Optimizer	Adam
Learning Rate	0.001
Activation Function	tanh
Loss function	MSE
Batch Size	1024
Epochs	10
Hidden Layers	2
Neurons per layer	32
Number of LSTM Units	64

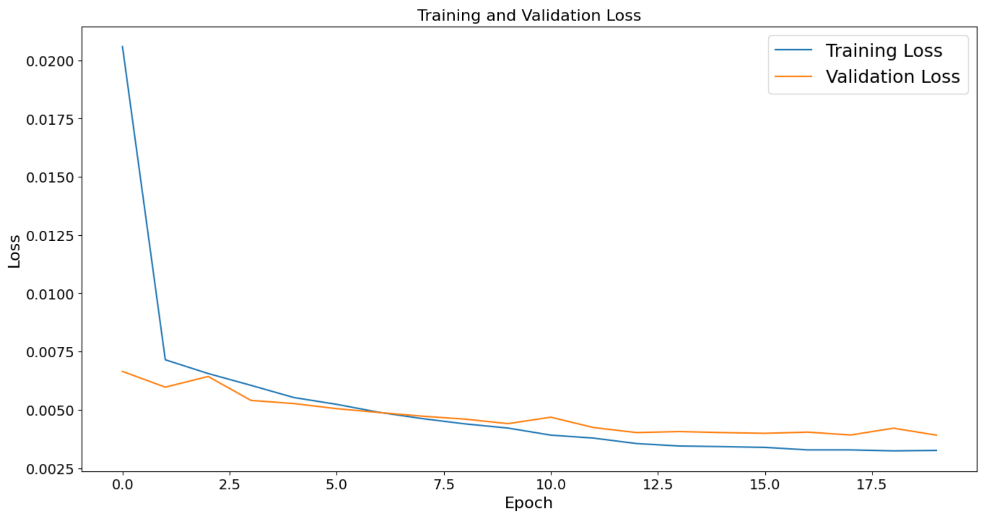


Figure 5.16: Loss function for training and validation data.

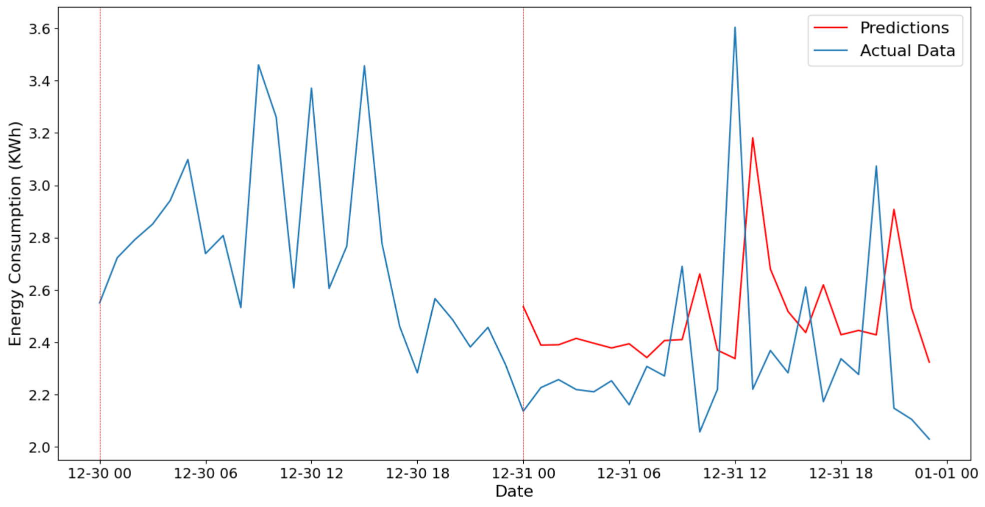


Figure 5.17: LSTM prediction.

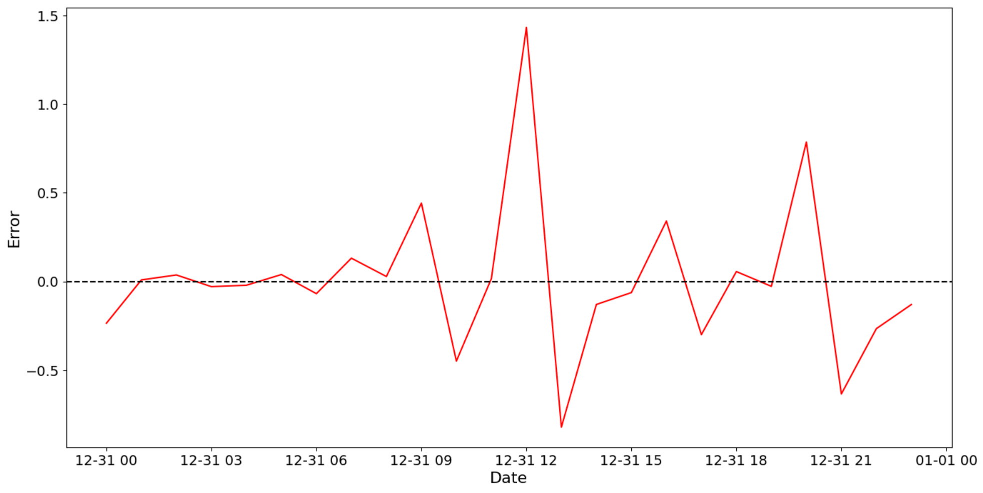


Figure 5.18: Error of LSTM method (test data).

5.7 Comparison Between Different Methods

The comparison of various approaches is depicted in Figure 5.19. It is evident from the illustration that PINN methods exhibit better accuracy in tracking the energy consumption pattern compared to other methods. This accuracy stems from the utilization of both physical parameters and historical energy consumption data in PINN methods, whereas the remaining approaches rely solely on historical data to identify energy consumption patterns in the household.

Comparing methods such as Physics-Informed Neural Networks (PINN), Long Short-Term Memory (LSTM), Seasonal Autoregressive Integrated Moving Average (SARIMA), and Prophet sheds light on their distinct characteristics. PINN combines physics principles and neural networks, providing clear insights and precise predictions, particularly in physical systems modeling. LSTM, a type of recurrent neural network (RNN), excels in capturing long-term dependencies in sequential data. SARIMA, a classical statistical method, skillfully models seasonal and trend components in time series data, providing reliable forecasts in various industries. While PINN, LSTM, and SARIMA focus on predicting patterns inherent in the data, Prophet primarily emphasizes capturing the trend component. Prophet's strength lies in its simplicity and ability to handle missing data and outliers effectively. This is why the SARIMA, LSTM, and PINNs are good at forecasting the pattern of data. By highlighting these distinctions, it becomes evident that while all methods aim to forecast future trends, their approaches and focuses vary, meeting diverse forecasting needs across different domains.

Table 5.5 provides Mean Absolute Error (MAE) values for various forecasting methods, including Physics-Informed Neural Networks (PINN) with specialized architectures (PhysReg and PhysNet), Long Short-Term Memory (LSTM) networks, Seasonal Autoregressive Integrated Moving Average (SARIMA) models, and the Prophet forecasting model. Lower MAE values indicate higher accuracy, with PINN (PhysReg) yielding the lowest MAE of 0.112, followed by PINN (PhysNet) at 0.136, LSTM at 0.188, SARIMA at 0.211, and Prophet at 0.302. These results suggest that Physics-Informed Neural Networks, particularly with a regular architecture, exhibit superior forecasting performance compared to other methods, highlighting their potential for accurate time series prediction tasks.

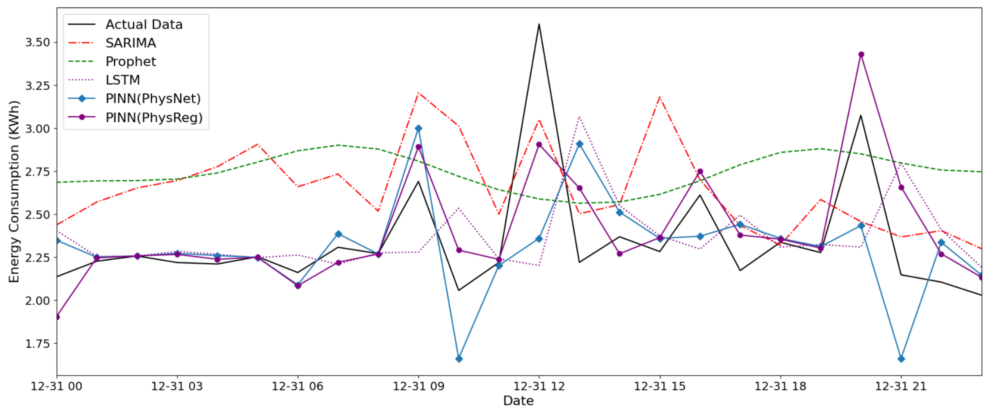


Figure 5.19: Comparison between different methods.

Table 5.5: MAE for different Methods (test data)

Method	MAE
PINN (PhysReg)	0.112
PINN (PhysNet)	0.136
LSTM	0.188
SARIMA	0.211
PROPHET	0.302

Chapter 6

Conclusion

In this study, we explored the predictive performance of five different time series forecasting models: SARIMA, Prophet, LSTM, and Physics-Informed Neural Networks (PINN), including PhysNet and PhysReg. These models were applied to a real-world dataset to forecast energy consumption in a building. Our analysis revealed interesting insights into the strengths and weaknesses of each model.

SARIMA, a classical statistical method, demonstrated its effectiveness in capturing the seasonal and trend components of the time series data. Its ability to incorporate past observations and seasonal patterns made it particularly suitable for forecasting tasks with clear cyclic patterns. However, SARIMA struggled to adapt to sudden changes or irregularities in the data, which limited its performance in scenarios with high volatility or non-linear trends.

Prophet provided a flexible and user-friendly framework for time series forecasting. It automatically handled issues such as seasonality, holidays, and outliers, making it easy to implement and interpret. Prophet excelled in capturing daily, weekly, and yearly seasonality patterns, making it a suitable choice for datasets with multiple seasonal components. However, its performance might degrade when dealing with irregular data patterns.

LSTM, a type of Recurrent Neural Network (RNN), demonstrated remarkable capabilities in capturing complex temporal dependencies within the data. Its ability to retain information over long sequences made it particularly effective in modeling non-linear and dynamic relationships in the time series data. LSTM models performed better than traditional statistical methods like SARIMA in scenarios characterized by high complexity and volatility. Nonetheless, LSTM models demand meticulous adjustment of hyperparameters and might encounter overfitting, particularly when dealing with restricted training datasets.

Physics-Informed Neural Networks (PINN) offered a unique approach by incorporating physical laws or constraints into the neural network architecture. This integration of domain knowledge improved the interpretability and generalization capabilities of the model, especially in scenarios where underlying physical principles govern the data generation process. However, the effectiveness of PINN heavily depends on the accuracy of the underlying physics-based constraints and the availability of domain expertise for model development.

In scenarios with clear seasonal patterns and limited volatility, `SARIMA` and Prophet models might often produce competitive results. However, for datasets with complex temporal dependencies or irregularities, `LSTM` and `PINN` models showed superior performance, with `LSTM` excelling in capturing long-term dependencies and `PINN` leveraging domain knowledge to improve forecast accuracy.

In conclusion, the choice of the most suitable forecasting model depends on various factors including the characteristics of the dataset, the forecasting horizon, and the availability of domain expertise. While `SARIMA` and `Prophet` offer robust solutions for capturing seasonal patterns and short-term forecasts, `LSTM` and `PINN` models provide more flexibility and accuracy in modeling complex temporal dynamics and long-term trends.

Bibliography

[1] M. Morari and J. H. Lee, 'Model predictive control: Past, present and future,' *Comput Chem Eng*, vol. 23, no. 4-5, pp. 667–682, May 1999. DOI: 10.1016/S0098-1354(98)00301-9.

[2] W. Mai and C. Y. Chung, 'Economic mpc of aggregating commercial buildings for providing flexible power reserve,' *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2685–2694, Sep. 2015. DOI: 10.1109/TPWRS.2014.2365615.

[3] C. Chen, J. Wang, Y. Heo and S. Kishore, 'Mpc-based appliance scheduling for residential building energy management controller,' *IEEE Trans Smart Grid*, vol. 4, no. 3, pp. 1401–1410, Sep. 2013. DOI: 10.1109/TSG.2013.2265239.

[4] C. R. Touretzky and M. Baldea, 'Integrating scheduling and control for economic mpc of buildings with energy storage,' *J Process Control*, vol. 24, no. 8, pp. 1292–1300, Aug. 2014. DOI: 10.1016/J.JPROCONT.2014.04.015.

[5] F. Oldewurtel *et al.*, 'Use of model predictive control and weather forecasts for energy efficient building climate control,' *Energy Build*, vol. 45, pp. 15–27, Feb. 2012. DOI: 10.1016/J.ENBUILD.2011.09.022.

[6] E. Zacekova, Z. Vana and J. Cigler, 'Towards the real-life implementation of mpc for an office building: Identification issues,' *Appl Energy*, vol. 135, pp. 53–62, Dec. 2014. DOI: 10.1016/J.APENERGY.2014.08.004.

[7] J. Ma, S. J. Qin and T. Salsbury, 'Application of economic mpc to the energy and demand minimization of a commercial building,' *J Process Control*, vol. 24, no. 8, pp. 1282–1291, Aug. 2014. DOI: 10.1016/J.JPROCONT.2014.06.011.

[8] T. Hilliard, L. Swan and Z. Qin, 'Experimental implementation of whole building mpc with zone based thermal comfort adjustments,' *Build Environ*, vol. 125, pp. 326–338, Nov. 2017. DOI: 10.1016/J.BUILDENV.2017.09.003.

[9] D. Sturzenegger, D. Gyalistras, M. Morari and R. S. Smith, 'Model predictive climate control of a swiss office building: Implementation, results, and cost–benefit analysis,' *IEEE Transactions on Control Systems Technology*, vol. 24, no. 1, pp. 1–12, Jan. 2016. DOI: 10.1109/TCST.2015.2415411.

[10] J. Seo, S. Kim, S. Lee, H. Jeong, T. Kim and J. Kim, 'Data-driven approach to predicting the energy performance of residential buildings using minimal input data,' *Build Environ*, vol. 214, Apr. 2022, Article 108911. DOI: 10.1016/J.BUILDENV.2022.108911.

[11] S. Zhan and A. Chong, 'Data requirements and performance evaluation of model predictive control in buildings: A modeling perspective,' *Renewable and Sustainable Energy Reviews*, vol. 142, May 2021, Article 110835. DOI: 10.1016/J.RSER.2021.110835.

[12] S. Yang, M. P. Wan, W. Chen, B. F. Ng and S. Dubey, 'Model predictive control with adaptive machine-learning-based model for building energy efficiency and comfort optimization,' *Appl Energy*, vol. 271, Aug. 2020, Article 115147. DOI: 10.1016/J.APENERGY.2020.115147.

[13] Y. Chen, M. Guo, Z. Chen, Z. Chen and Y. Ji, 'Physical energy and data-driven models in building energy prediction: A review,' *Energy Reports*, vol. 8, pp. 2656–2671, Nov. 2022. DOI: 10.1016/J.EGYR.2022.01.162.

[14] D. Picard, J. Drgoňa, M. Kvasnica and L. Helsen, 'Impact of the controller model complexity on model predictive control performance for buildings,' *Energy Build*, vol. 152, pp. 739–751, Oct. 2017. DOI: 10.1016/J.ENBUILD.2017.07.027.

[15] I. Antonopoulos *et al.*, 'Artificial intelligence and machine learning approaches to energy demand-side response: A systematic review,' *Renewable and Sustainable Energy Reviews*, vol. 130, Sep. 2020, Article 109899. DOI: 10.1016/J.RSER.2020.109899.

[16] H. Lee and Y. Heo, 'Simplified data-driven models for model predictive control of residential buildings,' *Energy Build*, vol. 265, Jun. 2022, Article 112067. DOI: 10.1016/J.ENBUILD.2022.112067.

[17] N.-T. Ngo, A.-D. Pham, T. T. H. Truong, N.-S. Truong and N.-T. Huynh, 'Developing a hybrid time-series artificial intelligence model to forecast energy use in buildings,' *Sci Rep*, vol. 12, no. 1, p. 15 775, Sep. 2022. DOI: 10.1038/s41598-022-19935-6.

[18] M. Bilgili and E. Pinar, 'Gross electricity consumption forecasting using lstm and sarima approaches: A case study of türkiye,' *Energy*, vol. 284, Dec. 2023, Article 128575. DOI: 10.1016/J.ENERGY.2023.128575.

[19] S. E. Sim, K. G. Tay, A. Huong and W. K. Tiong, 'Forecasting electricity consumption using sarima method in ibm spss software,' *Universal Journal of Electrical and Electronic Engineering*, vol. 6, no. 5B, pp. 103–114, Dec. 2019. DOI: 10.13189/ujeee.2019.061614.

[20] H. Kaur* and S. Ahuja, 'Sarima modelling for forecasting the electricity consumption of a health care building,' *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 12, pp. 2795–2799, Oct. 2019. DOI: `10.35940/ijitee.L2575.1081219`.

[21] P. J. Brockwell and R. A. Davis, Eds., *Introduction to Time Series and Forecasting*. New York, NY: Springer New York, 2002. DOI: `10.1007/b97391`.

[22] S. R. Riady and R. Apriani, 'Multivariate time series with prophet facebook and lstm algorithm to predict the energy consumption,' in *2023 International Conference on Computer Science*, IEEE: Information Technology and Engineering (ICCoSITE), Feb. 2023, pp. 805–810. DOI: `10.1109/ICCoSITE57641.2023.10127735`.

[23] M. Daraghmeh, A. Agarwal, R. Manzano and M. Zaman, 'Time series forecasting using facebook prophet for cloud resource management,' in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, Jun. 2021, pp. 1–6. DOI: `10.1109/ICCWorkshops50388.2021.9473607`.

[24] A. Banga and S. Sharma, 'Electricity demand forecasting models at hourly and daily level: A comparative study,' in *2022 International Conference on Advanced Computing Technologies and Applications (ICACTA)*, IEEE, Mar. 2022, pp. 1–5. DOI: `10.1109/ICACTA54488.2022.9753055`.

[25] R. J. Chadalavada, S. Raghavendra and V. Rekha, 'Electricity requirement prediction using time series and facebooks prophet,' *Indian J Sci Technol*, vol. 13, no. 47, pp. 4631–4645, Dec. 2020. DOI: `10.17485/IJST/v13i47.1847`.

[26] S. Chaturvedi, E. Rajasekar, S. Natarajan and N. McCullen, 'A comparative assessment of sarima, lstm rnn and fb prophet models to forecast total and peak monthly energy demand for india,' *Energy Policy*, vol. 168, Sep. 2022, Article 113097. DOI: `10.1016/J.ENPOL.2022.113097`.

[27] D. Durand, J. Aguilar and M. D. R-Moreno, 'An analysis of the energy consumption forecasting problem in smart buildings using lstm,' *Sustainability*, vol. 14, no. 20, p. 13 358, Oct. 2022. DOI: `10.3390/su142013358`.

[28] J. Jang, J. Han and S. B. Leigh, 'Prediction of heating energy consumption with operation pattern variables for non-residential buildings using lstm networks,' *Energy Build*, vol. 255, Jan. 2022, Article 111647. DOI: `10.1016/J.ENBUILD.2021.111647`.

[29] X. Wang, F. Fang, X. Zhang, Y. Liu, L. Wei and Y. Shi, 'Lstm-based short-term load forecasting for building electricity consumption,' *in*, vol. 2019 IEEE 28th International Symposium on Industrial Electronics (ISIE), IEEE, pp. 1418–1423, Jun. 2019. DOI: `10.1109/ISIE.2019.8781349`.

[30] I. Sulo, S. R. Keskin, G. Dogan and T. Brown, 'Energy efficient smart buildings: Lstm neural networks for time series prediction,' in *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*, IEEE, Aug. 2019, pp. 18–22. DOI: 10.1109/Deep-ML.2019.00012.

[31] J. Moon, S. Park, S. Rho and E. Hwang, 'A comparative analysis of artificial neural network architectures for building energy consumption forecasting,' *Int J Distrib Sens Netw*, vol. 15, no. 9, p. 155 014, Sep. 2019. DOI: 10.1177/1550147719877616.

[32] M. Fayaz, H. Shah, A. Aseere, W. Mashwani and A. Shah, 'A framework for prediction of household energy consumption using feed forward back propagation neural network,' *Technologies (Basel)*, vol. 7, no. 2, p. 30, Apr. 2019. DOI: 10.3390/technologies7020030.

[33] M. A. R. Biswas, M. D. Robinson and N. Fumo, 'Prediction of residential building energy consumption: A neural network approach,' *Energy*, vol. 117, pp. 84–92, Dec. 2016. DOI: 10.1016/j.energy.2016.10.066.

[34] S. Karatasou, M. Santamouris and V. Geros, 'Modeling and predicting building's energy use with artificial neural networks: Methods and results,' *Energy Build*, vol. 38, no. 8, pp. 949–958, Aug. 2006. DOI: 10.1016/j.enbuild.2005.11.005.

[35] J. Runge and R. Zmeureanu, 'Forecasting energy use in buildings using artificial neural networks: A review,' *Energies (Basel)*, vol. 12, no. 17, p. 3254, Aug. 2019. DOI: 10.3390/en12173254.

[36] D. L. Marino, K. Amarasinghe and M. Manic, 'Building energy load forecasting using deep neural networks,' *IECON*, vol. 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, IEEE, pp. 7046–7051, Oct. 2016. DOI: 10.1109/IECON.2016.7793413.

[37] M. Raissi, P. Perdikaris and G. Karniadakis, 'Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,' *Journal of Computational Physics*, vol. 378, pp. 686–707, Feb. 2019, ISSN: 00219991. DOI: 10.1016/j.jcp.2018.10.045.

[38] J. Huang, R. Qiu, J. Wang and Y. Wang, 'Multi-scale physics-informed neural networks for solving high Reynolds number boundary layer flows based on matched asymptotic expansions,' *Theoretical and Applied Mechanics Letters*, vol. 14, no. 2, p. 100 496, Mar. 2024, ISSN: 20950349. DOI: 10.1016/j.taml.2024.100496.

[39] S. Hou, X. Hao, D. Pan and W. Wu, 'Physics-informed neural network for simulating magnetic field of coaxial magnetic gear,' *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108 302, Jul. 2024, ISSN: 09521976. DOI: 10.1016/j.engappai.2024.108302.

[40] Q. Hou, Y. Li, V. P. Singh, Z. Sun and J. Wei, 'Physics-informed neural network for solution of forward and inverse kinematic wave problems,' *Journal of Hydrology*, vol. 633, p. 130 934, Apr. 2024, ISSN: 00221694. DOI: `10.1016/j.jhydrol.2024.130934`.

[41] J. D. Schmid, P. Bauerschmidt, C. Gurbuz, M. Eser and S. Marburg, 'Physics-informed neural networks for acoustic boundary admittance estimation,' *Mechanical Systems and Signal Processing*, vol. 215, p. 111 405, Jun. 2024, ISSN: 08883270. DOI: `10.1016/j.ymssp.2024.111405`.

[42] X. Qi, G. A. de Almeida and S. Maldonado, 'Physics-informed neural networks for solving flow problems modeled by the 2D Shallow Water Equations without labeled data,' *Journal of Hydrology*, p. 131 263, May 2024, ISSN: 00221694. DOI: `10.1016/j.jhydrol.2024.131263`.

[43] D. Zhang, L. Lu, L. Guo and G. E. Karniadakis, 'Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems,' *J. Comput. Phys.*, vol. 397, 2018. [Online]. Available: `https://api.semanticscholar.org/CorpusID:53368962`.

[44] S. Kabasi, A. L. Marbaniang and S. Ghosh, 'Gradient enhanced physics-informed neural network for iterative form-finding of tensile membrane structures by potential energy minimization,' *European Journal of Mechanics - A/Solids*, p. 105 332, Apr. 2024, ISSN: 09977538. DOI: `10.1016/j.euromechsol.2024.105332`.

[45] E. Vrettos, E. C. Kara, J. MacDonald, G. Andersson and D. S. Callaway, 'Experimental demonstration of frequency regulation by commercial buildings - part i: Modeling and hierarchical control design,' May 2016.

[46] G. Gokhale, B. Claessens and C. Develder, 'Physics informed neural networks for control oriented thermal modeling of buildings,' *Appl Energy*, vol. 314, May 2022, Article 118852. DOI: `10.1016/J.APENERGY.2022.118852`.

[47] J. Drgoňa, A. R. Tuor, V. Chandan and D. L. Vrabie, 'Physics-constrained deep learning of multi-zone building thermal dynamics,' *Energy Build*, vol. 243, Jul. 2021, Article 110992. DOI: `10.1016/j.enbuild.2021.110992`.

[48] G. E. P. Box, G. M. Jenkins, G. C. Reinsel and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley Sons, 2015.

[49] Suhartono, 'Time series forecasting by using seasonal autoregressive integrated moving average: Subset, multiplicative or additive model,' *J Math Stat*, vol. 7, pp. 20–27, Jan. 2011. DOI: `10.3844/jmssp.2011.20.27`.

[50] S. J. Taylor and B. Lethan, 'Forecasting at scale,' *PeerJ Prepr*, vol. 5, 2017.