CHALMERS
UNIVERSITY OF TECHNOLOGY

# AI - Driven Home Energy Management System for Profit and Grid Stability

Deep Reinforcement Learning and Predictive Models for Minimizing Peak Demand While Balancing Battery Degradation in a Dynamic Environment

Degree project report in Bachelor's Programme in Electrical Engineering

Adam Michelin

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Degree project report 2025

# AI - Driven Home Energy Management System for Profit and Grid Stability

Deep Reinforcement Learning and Predictive Models for Minimizing Peak Demand While Balancing Battery Degradation in a Dynamic Environment

Adam Michelin


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

AI - Driven Home Energy Management System for Profit and Grid Stability
Deep Reinforcement Learning and Predictive Models for Minimizing Peak Demand While Balancing Battery Degradation in a Dynamic Environment
Adam Michelin



Supervisor & Examiner: Thomas Hammarström, Department of Electrical Engineering, Chalmers University of Technology

Degree project report 2025
Department of Electrical Engineering Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Futuristic abstract home (Generated with Leonardo AI, Feb 2025)

Typeset in LaTeX
Gothenburg, Sweden 2025

AI - Driven Home Energy Management System for Profit and Grid Stability
Deep Reinforcement Learning and Predictive Models for Minimizing Peak Demand While Balancing Battery Degradation in a Dynamic Environment
Adam Michelin
Department of Electrical Engineering
Chalmers University of Technology

# Abstract

This thesis presents the development and implementation of an AI-driven home energy management system designed to optimize residential battery storage in response to Sweden's new power-based electricity tariffs, which introduce capacity fees based on avarage monthly power peaks starting January 2027. The system integrates three components: (1) multi-modal forecasting models for electricity prices, solar production, and household demand. (2) a Recurrent Proximal Policy Optimization (RPPO) reinforcement learning agent for real-time battery control and (3) automated orchestration via Prefect with Home Assistant integration. The forecasting stack (XGBoost and temporal convolutional networks (TCN)) achieves competitive accuracy, and the RL agent, trained on a custom reward balancing cost, solar utilization, and safety, learns price arbitrage and solar aware charging strategies. Field deployment on a 22 kWh battery with a 20 kW dual-orientation PV array demonstrates integration with real hardware and shows preliminary economic benefits under simulated seasonal conditions. The agent maintains 100% safety compliance (zero charge/discharge violations during final deployment) while achieving high grid independence. Although additional computational time for full training convergence and hyperparameter tuning remains as future work, these preliminary results underscore the strong potential of AI-driven residential energy management for cost savings and grid support.

Keywords: AI, RL, Home Energy Management System, Software.

# Acknowledgements

I would like to express my sincere gratitude to my supervisor and examiner, Thomas Hammarström, for his guidance and support throughout this project. His expertise and constructive feedback have been a big part in shaping this work.

I am deeply grateful to Chalmers University of Technology for providing the academic environment and resources necessary for this research. Special thanks to the Department of Electrical Engineering for their sponsorship.

This project would not have been possible without the incredible open-source community. I extend my appreciation to the developers and maintainers of the tools that formed the backbone of this work: Prefect for workflow orchestration, Gymnasium for reinforcement learning environments, XGBoost, PyTorch and TensorFlow for other machine learning implementations, and Home Assistant for smart home integration. Particular thanks to the forecast.solar team for providing reliable solar prediction services that enabled accurate PV forecasting.

Adam Michelin, Gothenburg, Jun 2025

# List of Acronyms

| | |
|---|---|
| AI | Artificial Intelligence |
| ANN | Artificial Neural Networks |
| API | Application Programming Interface |
| BMS | Battery Management System |
| CET | Central European Time |
| COP | Coefficient of Performance |
| DRL | Deep Reinforcement Learning |
| ETL | Extract, Transform, Load |
| EUPHEMIA | Pan-European Hybrid Electricity Market Integration Algorithm |
| GBT | Gradient-Boosted Trees |
| GRU | Gated Recurrent Unit |
| HEMS | Home Energy Management System |
| HMM | Hidden Markov Model |
| HTML | HyperText Markup Language |
| kW | Kilowatt |
| kWh | Kilowatt-hour |
| MAE | Mean Absolute Error |
| MDP | Markov Decision Process |
| ML | Machine Learning |
| PV | Photovoltaic |
| PVGIS | Photovoltaic Geographical Information System |
| $R^2$ | Coefficient of Determination |
| RESTful | Representational State Transfer |
| ROI | Return on Investment |
| RPPO | Recurrent Proximal Policy Optimization |
| SE3 | Swedish Electricity Price Zone 3 |
| SEK | Swedish Krona |
| SMOTE | Synthetic Minority Oversampling Technique |
| SoC | State of Charge |
| TCN | Temporal Convolutional Network |
| VAT | Value Added Tax |
| XGBoost | eXtreme Gradient Boosting |

# Contents

# List of Figures

# List of Tables

# 1
# Introduction

The transition towards renewable energy sources is reshaping residential electricity management. As Sweden implements power-based tariffs that charge consumers based on their highest monthly usage peaks, the need for intelligent home energy systems becomes critical. This shift in pricing, beginning in 2025, creates both a challenge and an opportunity for homeowners to actively manage their consumption patterns.

This thesis explores the development of an AI-driven home energy management system that leverages deep reinforcement learning to optimize battery storage operations. By intelligently coordinating solar production, battery charge/discharge cycles, and household consumption, the system aims to minimize electricity costs while contributing to grid stability through peak demand reduction. The work demonstrates how modern AI techniques can transform residential energy management from reactive rule-based approaches to adaptive, data-driven strategies that benefit both consumers and the broader electrical grid.

## 1.1 Background

The increasing integration of renewable energy sources, such as solar power, has introduced new challenges in energy management, particularly for residential and commercial buildings. Energy consumption patterns vary throughout the day, leading to peak demand periods that contribute to high electricity costs. Sweden is transitioning to an additional capacity-based fee system for households and small businesses. The new price model will be fully implemented by January 1, 2027, and some network operators like Ellevio, are introducing it earlier (January 2025). Their implementation bases a part of the monthly electricity bill on the users three highest monthly power peaks. We can assume that other operators will implement a similar pricing model [1], [2].

By incentivizing households to reduce their peak power consumption through capacity based fees, Sweden aims to flatten the aggregate demand curve across the grid. When many homes shift their energy usage away from peak periods, this collective behavior reduces stress on transmission infrastructure, minimizes voltage fluctuations, and decreases the need for expensive grid reinforcements. This creates a win-win scenario where

consumers save money while contributing to a more resilient and stable energy system addressing the core reason why Sweden is transitioning to this new tariff structure [9].

Home battery systems offer a viable solution for mitigating peak electricity demand, enhancing overall energy efficiency, and potentially generating income through participation as a micro-producer. By storing excess energy from solar panels or the grid during off-peak hours, these batteries can supply power when demand is high, reducing reliance on expensive grid electricity. However, optimizing the use of home batteries presents several challenges, including balancing energy storage, predicting peak demand, and minimizing battery degradation over time.

A well-optimized home energy system can lead to lower electricity costs for consumers while contributing to a more balanced and efficient power grid [3]. This represents a win-win solution for both users and electricity providers, as the primary goal of power-based tariffs is to encourage more even energy distribution and reduce grid strain [2].

Artificial intelligence (AI) and machine learning techniques, particularly reinforcement learning (RL), have shown significant promise in optimizing energy usage [6]. These methods can dynamically adjust battery charging and discharging strategies based on real-time data, predictive models, and user preferences. While traditional battery management systems rely on predefined rules, there is growing interest in adaptive, data-driven approaches that can respond dynamically to changing electricity prices and user behavior [3].

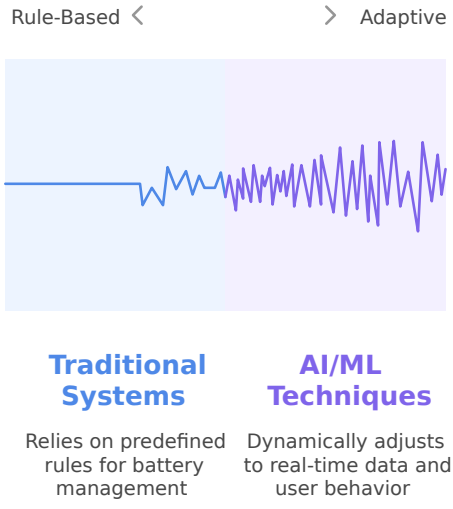



**Figure 1.1:** Rule-based vs adaptive control: fixed rules (left) versus data-driven adjustments (right).

## 1.2 Purpose

The purpose of this thesis is to **design**, **implement**, and **experimentally evaluate** and AI-driven battery optimization system that **minimizes electricity costs** while **supporting grid stability**. With Sweden transitioning to a power-based tariff system, residential homes and small businesses are forced to strategically manage their energy consumption to avoid high costs.

More concretely, the work seeks to:

I **Integrate a full stack optimization platform**, comprising multi-model forecasting (**demand**, **spot prices**, and **solar production**) together with a Recurrent PPO controller into a real residential installation built around a 22 kWh battery and a dual-orientated 20 kW PV array.

II **Automate end-to-end orchestration** of data ingestion, model retraining and control dispatch through a Prefect server workflow pipeline, thereby demonstrating a production-grade architecture that can run unattended. The solution will be integrated with Home Assistant for real-time monitoring and automation, providing a practical and user-friendly implementation of AI-powered energy management.

III **Quantify economic and technical impact** by comparing the AI system against rule-based and static-schedule baselines, though the project timeline will not be able to cover monthly trials a general system performance with respect to metrics will be made by simulation different systems.

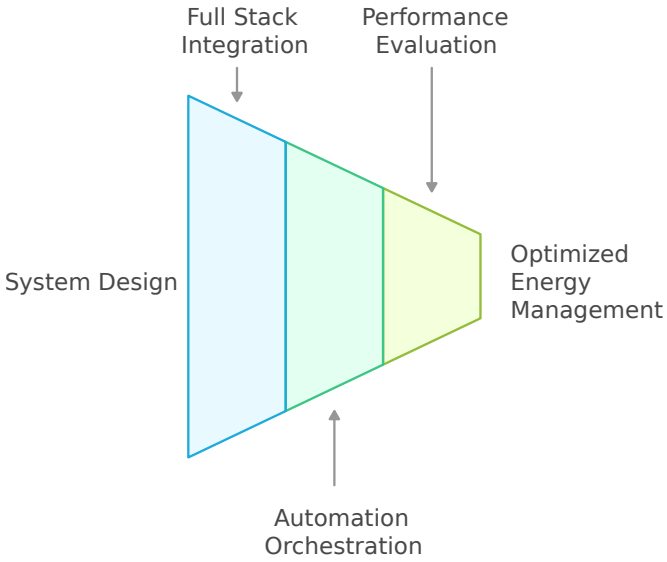



**Figure 1.2:** System purpose diagram

## 1.3 Goals

To translate the overarching purpose into tangible outcomes, the thesis aims for six concrete goals:

I **Establish a solid theoretical foundation** by researching the state of the art in AI-based home energy management and Sweden's power-based tariff schemes.

II **Develop accurate forecasting modules** for household demand, day-ahead spot prices, and photovoltaic production to supply the controller with reliable short-term predictions.

III **Design, train, and validate a reinforcement-learning controller** that converts forecasts, battery constraints, and tariff rules into battery charge/discharge actions.

IV **Deploy the complete optimization platform** in a real residential setting, integrating hardware and software for continuous, unattended operation while providing an intuitive user interface.

V **Evaluate practical performance and limitations** through a combination of field trials and simulation studies, benchmarking the AI system against rule-based and static-schedule baselines.

VI **Publish an open reference implementation** source code, configuration, and documentation to facilitate replication and future research.

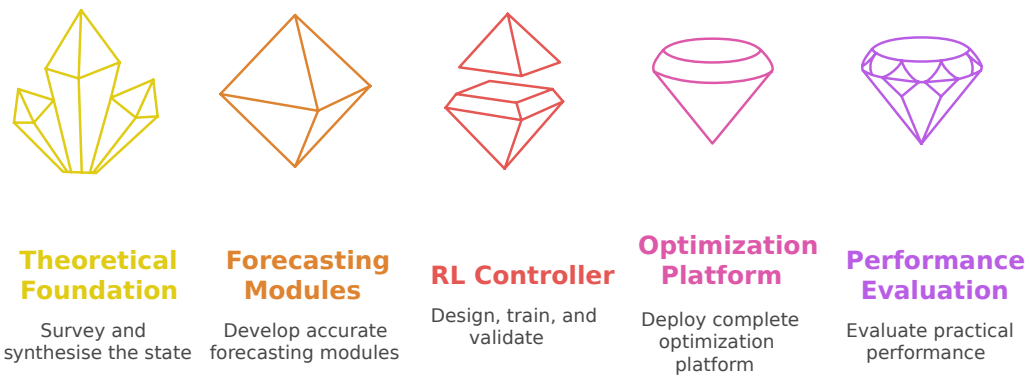



**Figure 1.3:** System goal diagram

## 1.4 Limitations / Demarcations

While the work aspires to provide an end-to-end proof-of-concept, several deliberate boundaries have been set to keep the project feasible within a bachelor-level time frame:

I **Single-site pilot**. All field experiments are conducted in one house located in Stockholm, equipped with a 22kWh battery and a 20kW dual-orientation PV array. Results may not generalise to other climates, tariff zones, hardware configurations, or user patterns.

II **Evaluation horizon**. Because of time constraints, the RL controller is validated over a couple of days. Long-term seasonal effects and battery ageing beyond this window are assessed only through simulation.

III **Battery-centric control**. The controller schedules only battery charge and discharge. Flexible loads such as EV charging, heat pumps, and smart appliances are left unmanaged. While the codebase supports future integration, safe deployment would require additional safety logic and hyperparameter tuning that lie outside this study.

IV **Simplified battery degradation model**. Cycle life is approximated by an energy-throughput penalty calibrated and calculated on the specific battery installed; electrochemical ageing mechanisms (temperature, C-rate, SoC swing) are not explicitly modelled.

V **Fixed 15-minute control granularity**. The decision interval operates on seemingly low resolution but is choosen for simplicity. Unforseen demand spikes are safely handled with a buffer script to ensure proper home energy managment safety. Faster RL dynamics remain outside the study scope.

VI **Hardware and network reliability**. The prototype runs on a server located at the home owner. Cyber-security hardening via HTTPS and ZeroTier [1] tunneling are in place to ensure safe integration together with basic crash-handling since.

The above demarcations ensure that the thesis remains achievable while still demonstrating the viability of an AI-driven battery optimizer under Sweden's forthcoming power-based tariff regime. Future research can ease these constraints to address a more sophisticated and broader system

[1] ZeroTier is a networking solution to securely connect virtual networks across various devices and locations.

# 2

# Theory

This chapter lays the theoretical groundwork on which the remainder of the thesis is built. It opens with Sweden's emerging power-based tariff design and details how the new capacity fee formula based on the three highest hourly peaks, reshapes the household cost landscape section 2.1.

Section 2.2 then translates the physical installation into mathematics: grid-exchange balances, battery state-of-charge dynamics, and a tilt-aware photovoltaic model that links solar geometry to electrical output.

With the energy flows formalised, section 2.3 surveys the machine-learning tools that will later drive optimisation: gradient-boosted trees, temporal convolutional networks, LSTMs, and their integration into a Recurrent PPO reinforcement-learning framework.

Finally, section 2.4 establishes the accuracy, cost-saving, and robustness metrics that will benchmark both forecasts and control policies throughout the study. Together these four sections provide the analytical scaffold required to understand the methods and results that follow.

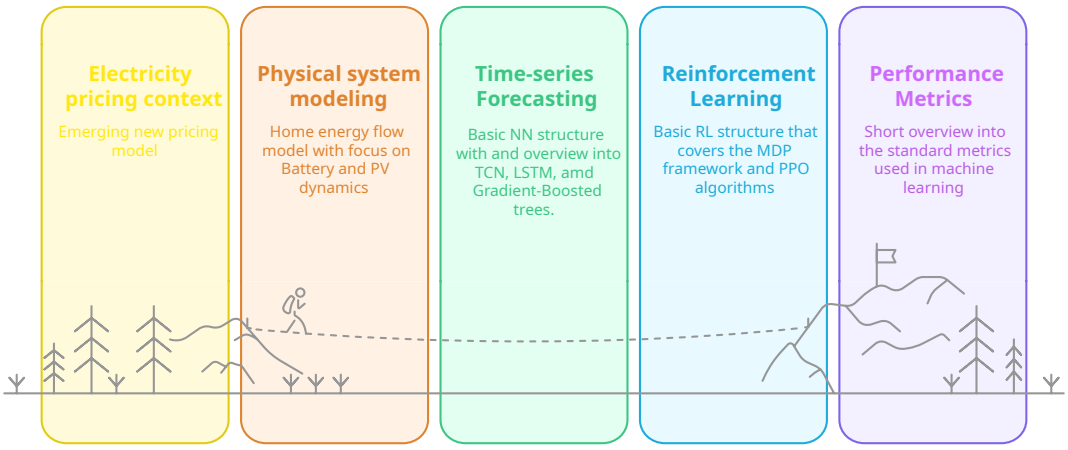



**Figure 2.1:** Theory chapter overview

## 2.1 Electricity–pricing context

The monthly cost for a household customer is formalised in Equation (2.1), with all symbols defined in Table 2.1 At its core, this expression aggregates four main components:

(i). The per-kWh energy price set by Nord Pool's day-ahead market.

(ii). A per-kWh variable grid-energy charge imposed by the DSO.

(iii). A fixed monthly grid fee covering metering and service costs.

(iv). From 2025 onwards, a capacity fee based on peak power consumption [1].

The energy-based terms are multiplied by a uniform VAT rate [8], while the capacity fee mandated under Ei's EIFS 2022:1 regulation charges customers for their three highest hourly averages each month, thereby incentivising lower peaks [9].

To further encourage off-peak usage, many DSOs (e.g. Ellevio) apply a 50% discount on any registered peak between 22:00-06:00, effectively halving those hours when calculating the average of the top three peaks. This structure strongly signals to consumers that shifting energy usage such as EV or battery charging to nighttime hours reduces their capacity charges, aligning household behaviour with grid-stability goals [5].

**Table 2.1:** Notation for the Swedish retail-electricity price model

| Symbol | Meaning | Unit |
|---|---|---|
| $\lambda_t$ | Nord Pool spot price in hour $t$ | öre kWh$^{-1}$ |
| $\tau_E$ | Energy-tax rate (ex-VAT) | öre kWh$^{-1}$ |
| $\phi$ | Variable grid-energy charge (ex-VAT) | öre kWh$^{-1}$ |
| $v$ | VAT rate (= 1.25) | - |
| $E_t$ | Energy consumed in hour $t$ | kWh |
| $\beta$ | Capacity-fee unit price | SEK kW$^{-1}$ |
| $P_i$ | $i$-th largest hourly mean power in $\mathcal{M}$ | kW |
| $w_t$ | 0.5 during 22:00 to 05:59, 1 otherwise | - |
| $C_{\text{fix}}$ | Fixed monthly grid fee | SEK |

$$C_{\text{month}} \;=\; v \sum_{t \in M} (\lambda_t + \phi + \tau_E)\, E_t \;+\; C_{\text{fix}} \;+\; \beta\, \frac{w_t (P_{(1)} + P_{(2)} + P_{(3)})}{3}. \tag{2.1}$$

## 2.2 Physical system modeling

A precise mathematical power flow model is essential for linking control actions to cost outcomes. Grid exchange at time $t$ is determined by Equation(2.2) where $L_t$ is the household load, $P_t^{\mathrm{PV}}$ the PV output, and $P_t^{\mathrm{dis}}/P_t^{\mathrm{ch}}$ the battery discharge/charge power. Positive $P_t^{\mathrm{grid}}$ denotes import whereas a negative value relates to exported power. Figure 2.2 illustrates these flows in the live Home Assistant dashboard.

$$P_t^{\mathrm{grid}} = L_t - P_t^{\mathrm{PV}} - P_t^{\mathrm{dis}} + P_t^{\mathrm{ch}}, \tag{2.2}$$

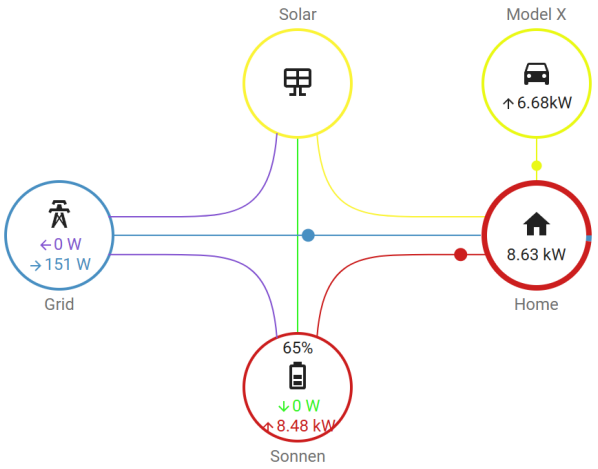



**Figure 2.2:** Real-time energy-flow dashboard in Home Assistant.

### 2.2.1 Battery State-of-Charge (SoC) Dynamics

The battery dynamic is calculated by the round-trip efficiency shown in Equation (2.3). It quantifies the energy returned by a battery versus the energy put in, accounting for losses from heat, internal resistance, and inverter conversion. This means that the battery must discharge slightly more energy than the load actually receives (Fig. 2.3)

$$E_{t+1} = E_t + \underbrace{\eta_{\mathrm{ch}}\, P_t^{\mathrm{ch}}\, \Delta t}_{\text{net energy stored}} - \underbrace{\frac{P_t^{\mathrm{dis}}\, \Delta t}{\eta_{\mathrm{dis}}}}_{\text{energy withdrawn}} \tag{2.3}$$

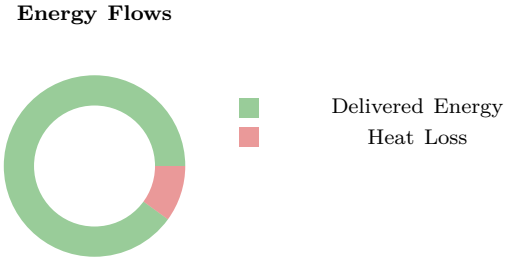



**Figure 2.3:** Symbolic proportion of delivered energy versus heat losses.

### 2.2.2 Photovoltaic Dynamics

**Table 2.2:** Notation for symbols and parameters used in photovoltaics

| Symbol | Meaning | Unit |
|---|---|---|
| $\delta$ | Solar declination angle (tilt north/south of equator) | rad |
| $\phi$ | Site latitude (positive north of equator) | rad |
| $\beta$ | Panel tilt angle (from horizontal) | rad |
| $\gamma$ | Panel azimuth angle (deviation from poles) | rad |
| $H$ | Hour angle (solar time offset: $H = 0$ at solar noon) | rad |
| $\theta_i$ | Solar incidence angle (between sun-rays and panel normal) | rad |
| $G(\tau)$ | Plane-of-array irradiance at time $\tau$ | $\mathrm{kW\,m^{-2}}$ |
| $R_t$ | Raw radiation sum over $[t-\Delta t, t]$ | $\mathrm{kWh\,m^{-2}}$ |
| $R_t^{\mathrm{eff}}$ | Cosine-corrected radiation sum over $[t-\Delta t, t]$ | $\mathrm{kWh\,m^{-2}}$ |
| $A_{\mathrm{PV}}$ | Effective PV array area | $\mathrm{m^2}$ |
| $\eta_{\mathrm{PV}}$ | PV conversion efficiency | – |
| $P_t^{\mathrm{PV}}$ | Average PV electrical power over interval | kW |

Photovoltaic output is modeled from the total solar radiation over each control interval $\Delta t$ [11]. First define the raw radiation sum:

$$R_t = \int_{t-\Delta t}^{t} G(\tau)\,\mathrm{d}\tau \quad \left[\mathrm{kWh\,m^{-2}}\right], \tag{2.4}$$

where $G(\tau)$ is the plane-of-array irradiance at time $\tau$ for hourly steps $\Delta t$.

To account for the panel's tilt and orientation, we apply a cosine-law correction. Let

$$R_t^{\mathrm{eff}} = \int_{t-\Delta t}^{t} G(\tau)\,\cos\Big(\theta_i(\tau)\Big)\,\mathrm{d}\tau, \tag{2.5}$$

where $\theta_i(\tau)$ is the solar incidence angle defined below and shown in 2.7(b).

If the PV array has effective area $A_{\mathrm{PV}}$ [$\mathrm{m^2}$] and module efficiency $\eta_{\mathrm{PV}}$ treated as a constant for simplicity[1], the interval averaged electrical power is

$$P_t^{\mathrm{PV}} = \eta_{\mathrm{PV}}\,A_{\mathrm{PV}}\,\frac{R_t^{\mathrm{eff}}}{\Delta t} \quad \Big[\mathrm{kW}\Big]. \tag{2.6}$$

With the usual $\Delta t = 1\,\mathrm{h}$, this reduces to $P_t^{\mathrm{PV}} = \eta_{\mathrm{PV}} A_{\mathrm{PV}} R_t^{\mathrm{eff}}$, i.e. power is directly proportional to the cosine-corrected hourly radiation.

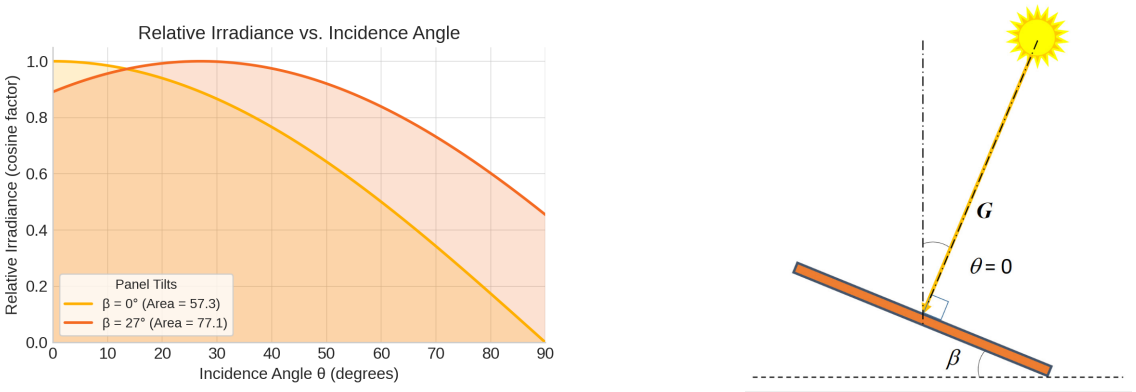
[1] $\eta_{PV}$ depends partly on temperature, spectral effects, and long-term degradation among other.

For a fixed panel tilt $\beta$ and azimuth $\gamma$, the solar incidence angle on the module surface is shown in Equation (2.7) [13]. The effective plane-of-array irradiance is then

$$G^{\text{eff}}(t) = G(t)\cos\Big(\theta_i(t)\Big).$$

Figure 2.7(a) shows the relative factor $\cos(\theta_z - \beta)$ for $\beta = 27°$ versus $\beta = 0°$, highlighting that tilt increases the daily integral of $G^{\text{eff}}$ by approximately 35 %. Figure 2.7(b) illustrates the basic cosine geometry.

$$\begin{aligned}\theta_i(t) = \cos^{-1}\Big[&\sin\delta\,\sin\phi\,\cos\beta - \sin\delta\,\cos\phi\,\sin\beta\,\cos\gamma \\ &+ \cos\delta\,\cos\phi\,\cos\beta\,\cos H + \cos\delta\,\sin\phi\,\sin\beta\,\cos\gamma\,\cos H \\ &+ \cos\delta\,\sin\beta\,\sin\gamma\,\sin H\Big], \quad (2.7)\end{aligned}$$




**(a)** **(b)**

**Figure 2.4:** (a) Relative irradiance reduction $\cos(\theta_z - \beta)$ as a function of incidence angle for two fixed tilts: horizontal ($\beta = 0°$) and a south-facing roof pitch ($\beta = 27°$). (b) Geometric interpretation of the cosine law [12].

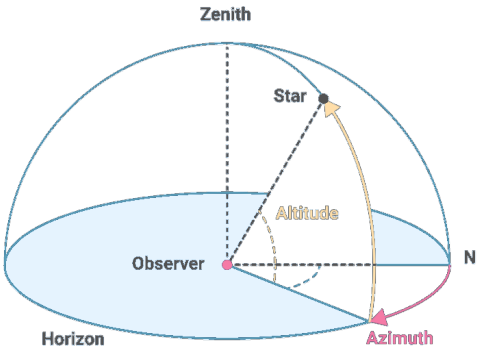



**Figure 2.5:** Horizontal (altitude–azimuth) coordinate system used in solar-position calculations.

## 2.3 Machine Learning

Machine learning (ML) is a subfield of artificial intelligence focused on developing algorithms that automatically infer patterns, relationships, and decision rules from data, rather than relying on strict hard coded rules. Over the past decade, advances in statistical learning theory, optimization methods, and scalable computing have enabled ML models to tackle increasingly complex tasks, from image classification to sequential decision-making.

In energy systems, the high dimensionality and nonlinearity of phenomena such as fluctuating demand, variable renewable generation, and time-varying market prices pose challenges for conventional rule-based controllers. ML approaches address these challenges by learning functional mappings directly from historical and real-time measurements, adapting to changing conditions, and continually improving as more data become available. Broadly speaking, ML methods can be divided into:

- **Supervised learning:** Models trained on labeled input–output pairs to predict quantities of interest (e.g. load forecasting, price forecasting).
- **Unsupervised learning:** Techniques that discover structure in unlabeled data (e.g. clustering of consumption patterns, anomaly detection).

Each class offers distinct capabilities, supervised models excel at probabilistic forecasts, unsupervised methods reveal hidden structure, In this chapter, we focus on the two ML components driving our AI-based energy optimizer: (1) time-series forecasters for generating multi-step-ahead predictions of electrical load, PV output, and spot prices, and (2) a deep reinforcement-learning agent that selects battery charge/discharge actions. The following sections outline the essential concepts and mathematical formulations underlying each model class before delving into their specific architectures and training algorithms.

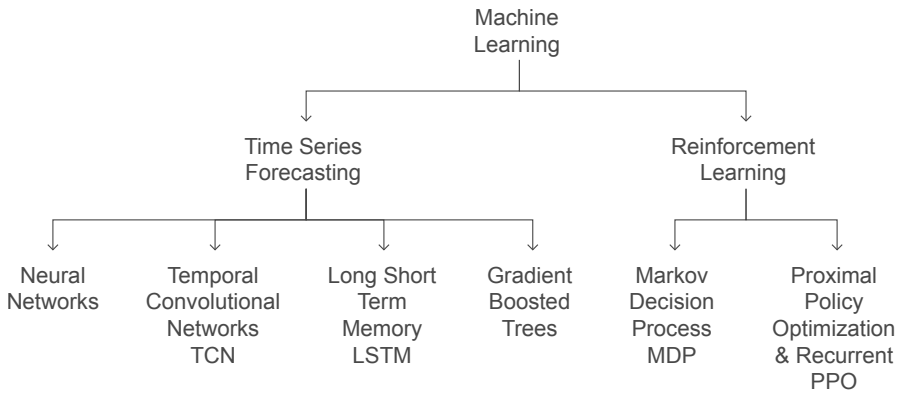



**Figure 2.6:** Machine learning chapter overview showing the difference branches between Forecasting and Reinforcement learning classes

### 2.3.1 Time-series forecasting

Time series forecasting is a predictive modeling technique that analyzes *past sequential data points over time* to identify patterns and trends that can be extrapolated to make predictions see Fig 2.7b. This differs from other modeling techniques like tabular forecasting that treats each time series as an independent example. We define time-series forecasting mathematically by letting

$$\mathbf{x}_{1:t} = \{x_1, \ldots, x_t\}$$

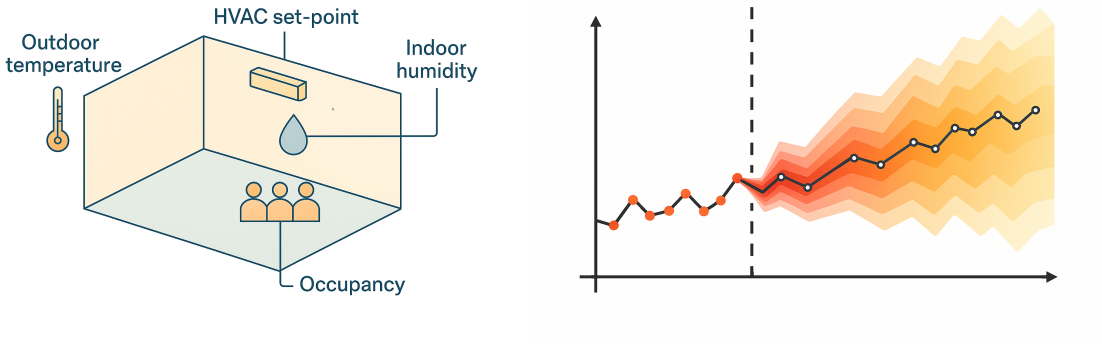
denote a history and $\hat{x}_{t+h}$ its $h$-step prediction.

Naturally, more accurate and robust forecasts can be obtained by incorporating multiple coherent data sources rather than relying solely on past target values. For a simple real-world example: forecasting the indoor temperature of an office room, one might include the historical indoor temperature series $\mathbf{x}_{1:t}$ together with exogenous inputs, lets call this

$$\mathbf{z}_{1:t} = \Big\{\text{outdoor temp},\ \text{occupancy},\ \text{HVAC set-point},\ \text{indoor humidity}\Big\}_{1:t}.$$

see Fig 2.7a. A general multivariate forecasting function then takes the form

$$\hat{x}_{t+h} \;=\; f\Big(x_{1:t},\ \mathbf{z}_{1:t}\Big),$$




**(a)** **(b)**

**Figure 2.7:** (a) Inputs for forecasting indoor temperature: outdoor temp, occupancy, HVAC set-point, indoor humidity. (b) Example $h$-step-ahead forecast.

Building on the general forecasting formulation introduced above, we now turn to a powerful class of nonlinear models capable of capturing complex, non-stationary relationships in time-series data: neural networks.

#### 2.3.1.1 Neural Networks (NN)

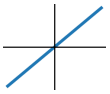
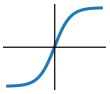
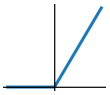
Neural networks are a family of nonlinear function approximators inspired by the structure of biological brains. The simplest unit is the *perceptron*, which computes a weighted sum of its inputs plus a bias,

$$net = \sum_{i=0}^{n} w_i x_i + b, \tag{2.8}$$

$$o = \sigma(net), \tag{2.9}$$

where $\sigma(\cdot)$ is an activation function (e.g. sigmoid, ReLU, see Table 2.3) that introduces nonlinearity. By stacking many such units into layers, we obtain a *multilayer perceptron* (MLP) see Figure 2.8,

**Table 2.3:** Common activation functions

| Plot | Function name | Definition |
|---|---|---|
|  | Unit step | $\sigma_{\text{step}}(net) = \begin{cases} 1 & net > 0, \\ -1 & \text{otherwise.} \end{cases}$ |
|  | Linear | $\phi_{\text{lin}}(net) = net$ |
|  | Logistic (Sigmoid) | $\sigma_{\log}(net) = \dfrac{1}{1+e^{-net}}$ |
|  | Hyperbolic tangent | $\sigma_{\tanh}(net) = \dfrac{e^{net}-e^{-net}}{e^{net}+e^{-net}}$ |
|  | Rectified Linear Unit (ReLU) | $\sigma_{\text{ReLU}}(net) = \max(0, net)$ |

Activation functions are a core component of neural networks, introducing the nonlinearity that enables these models to learn complex patterns beyond simple linear mappings. Early neural architectures predominantly used the logistic sigmoid function, which "squeezes" the weighted sum into the interval (0, 1), mirroring the biological concept of a neuron being inactive or active. However, as networks have grown deeper and more complex, the Rectified Linear Unit (ReLU) function as it closely mimics the biological neuron's threshold behavior, where an output "fires" only after the input surpasses a certain level and dramatically accelerates training convergence [14].

Training a neural network entails finding the set of weights $W^{(\ell)}$ and biases $b^{(\ell)}$ for each layer $\ell = 1, \ldots, L$ that minimize a chosen loss function $\mathcal{J}(\theta)$. For regression tasks one typically uses the mean-squared error,

$$\mathcal{J}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \Big( y_i - f_\theta(\mathbf{u}_i) \Big)^2,$$

where $\theta = \{W^{(\ell)}, b^{(\ell)}\}$ collectively denotes all parameters, $\mathbf{u}_i$ the $i$th input, and $y_i$ its target. Computing the gradient $\nabla_\theta \mathcal{J}$ efficiently for deep, layered models is made possible by *backpropagation*, which applies the chain rule to propagate error signals from the output layer back to each parameter. Parameters are then updated iteratively.

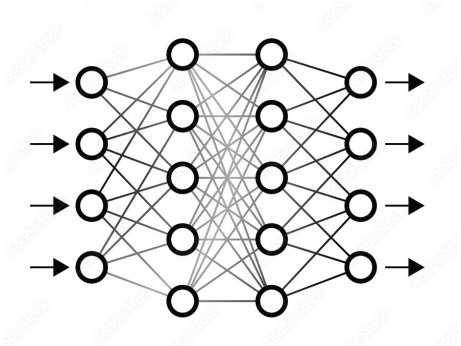


**Figure 2.8:** Multilayer perceptron a kind of neural network

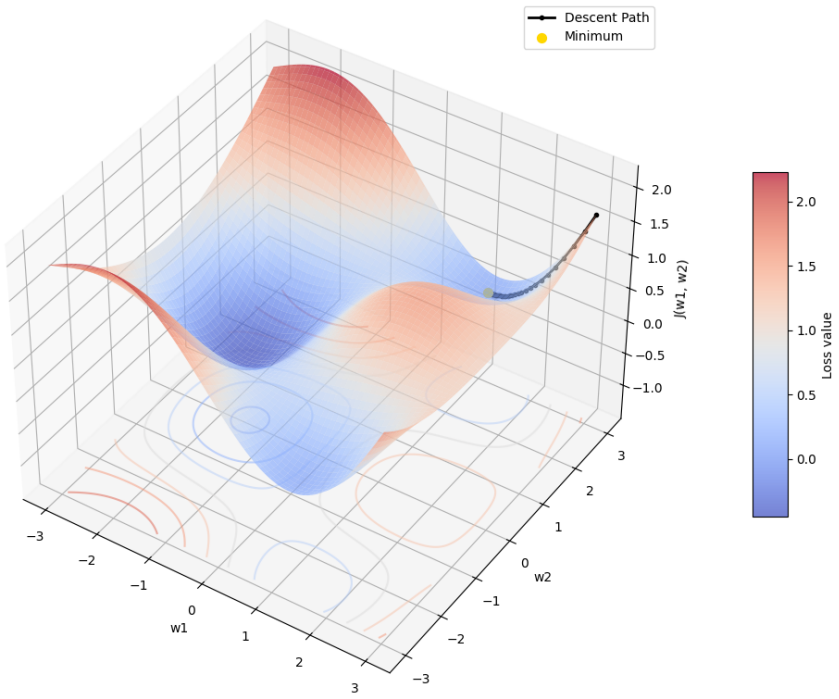



**Figure 2.9:** Gradient Descent to Global Minimum on Loss Surface

#### 2.3.1.2 Temporal Convolutional Networks (TCN)

Temporal Convolutional Networks (TCNs) are convolutional architectures specifically designed for sequence data. Unlike recurrent models, TCNs use 1D causal convolutions filters that only span current and past time steps so that at no point does information "leak" from the future. By stacking layers with exponentially increasing dilation (spacing between filter taps), a TCN achieves a very large receptive field with relatively few layers, allowing it to capture long-range dependencies efficiently.

Each convolutional layer is wrapped in a residual block: after applying a small stack of operations (convolution, weight normalization, ReLU and dropout), the block's input is added back to its output. These skip-connections stabilize training and help gradients flow through deep networks. In practice, TCNs combine the parallelism of CNNs with the sequential modeling power of RNNs, often outperforming LSTM/GRU on large forecasting benchmarks.

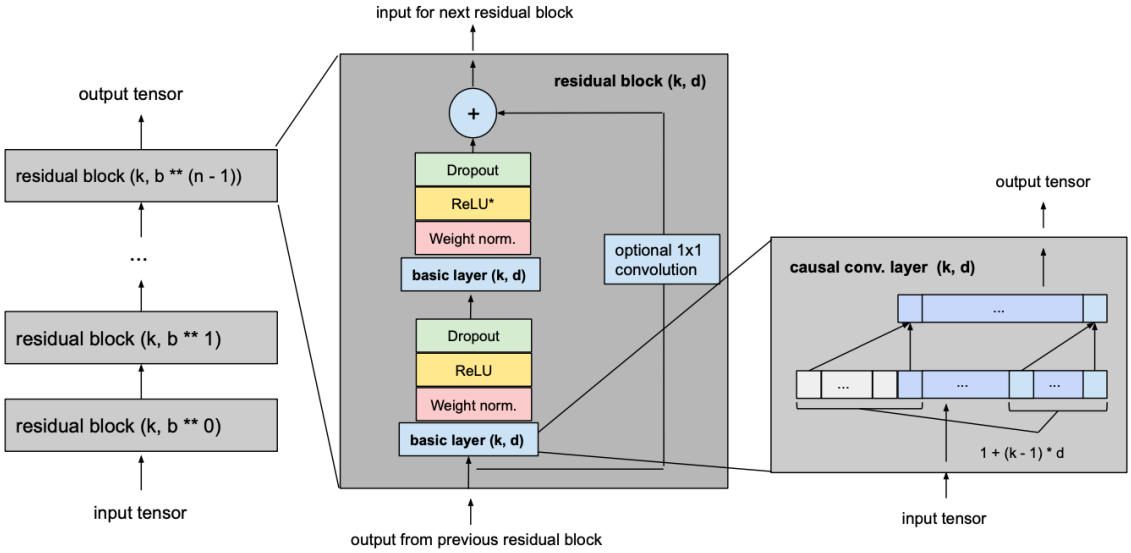



**Figure 2.10:** TCN model with equal to $input_{length}$, $k$ equal to $kernel_{size}$, $b$ equal to $dilation_{base}$, $k \geq b$ and with a minimum number of residual blocks for full history coverage $n$, where $n$ can be computed from the other values as explained above Source

#### 2.3.1.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory networks (LSTMs) are a type of recurrent neural network [15] designed to capture long-range dependencies in sequence data while avoiding the vanishing and exploding gradient problems of standard RNNs. At their core, each LSTM cell maintains a *cell state* $C_t$, which carries information forward unchanged unless explicitly modified by learned gating mechanisms. Three gates regulate this flow:

- **Forget gate** $f_t$ decides which information from the previous cell state $C_{t-1}$ to discard.
- **Input gate** $i_t$ determines which new information to add to the cell state, often computed as a candidate update $\tilde{C}_t$.
- **Output gate** $o_t$ controls which parts of the updated cell state are exposed as the hidden state $h_t$.

By learning when to remember, update, or output information, LSTMs excel at modeling time-series with complex temporal patterns, such as seasonality, trends, and irregular events, making them a cornerstone architecture for sequential forecasting tasks[15].

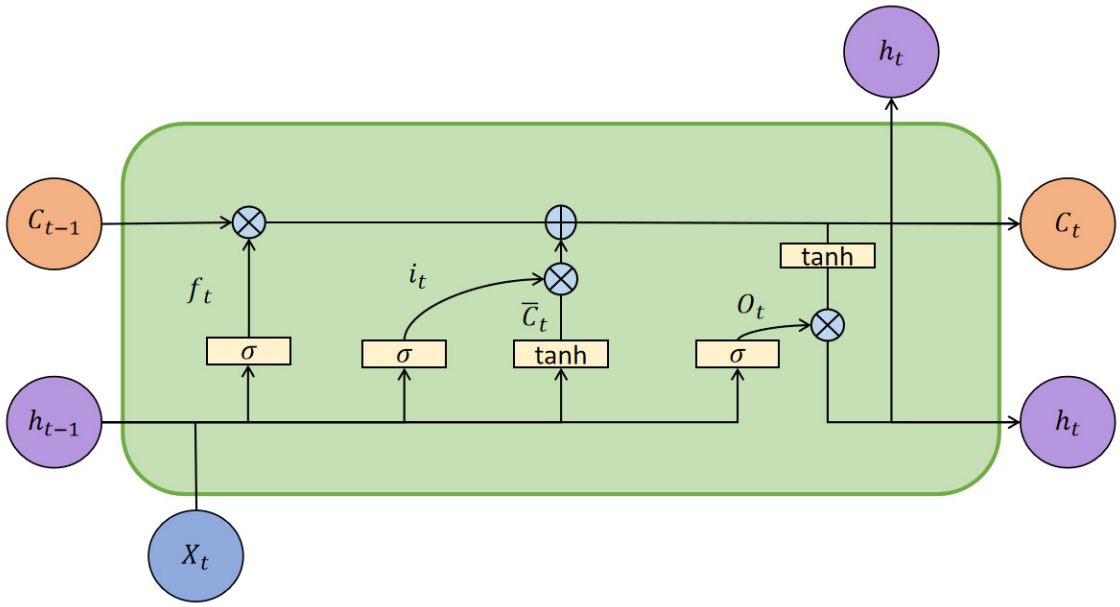



**Figure 2.11:** Internal structure of an LSTM cell, showing how the forget gate $f_t$, input gate $i_t$, and output gate $o_t$ regulate the flow of information into and out of the cell state $C_t$ and hidden state $h_t$.

#### 2.3.1.4 Gradient-Boosted Trees

Gradient-Boosted Trees (GBT) are a powerful ensemble method that builds a strong predictive model by sequentially adding decision trees, each one trained to correct the mistakes of the ensemble so far. Rather than fitting one large, complex tree, GBT constructs many small "weak learners" (shallow trees), combining them into a robust predictor. Key characteristics include:

- **Stage-wise learning:** Each new tree focuses on the residual errors differences between the true target and the current model's predictions and attempts to reduce them.
- **Shrinkage and learning rate:** A small scaling factor (learning rate) limits each tree's impact, encouraging gradual improvements and reducing overfitting.
- **Tree complexity control:** Maximum depth, minimum child weight, and subsampling ratios govern each tree's size and the fraction of data used, providing regularization.
- **Handling missing and categorical data:** Modern GBT implementations (e.g. XGBoost) automatically learn optimal "default directions" for missing values, and efficiently bin or one-hot encode categorical features.
- **Feature importance and interpretability:** By measuring how often and how effectively features split the data, GBTs offer built-in diagnostics to rank predictors and guide feature engineering.
- **Scalability and speed:** Highly optimized libraries exploit parallel tree construction, out-of-core computation, and low-level optimizations to handle large-scale datasets.

In time-series forecasting, GBT models excel when combined with thoughtfully engineered features lagged values, rolling statistics (mean, variance), and calendar indicators (hour of day, day of week). Their ability to capture nonlinear interactions without extensive hyperparameter tuning makes them a popular baseline and often a component in hybrid deep-learning ensembles.

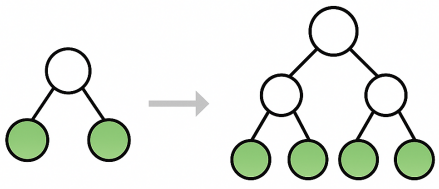


**Figure 2.12:** Illustration of gradient boosting: successive shallow trees are added to the ensemble, each one correcting the residual errors of the accumulated model.

### 2.3.2 Deep Reinforcement learning

Deep reinforcement learning augments the standard RL framework by representing policies $\pi_\theta(a \mid s)$ and value functions $V^\pi(s)$, $Q^\pi(s,a)$ with deep neural networks parameterized by $\theta$. Instead of tabular lookup, the network takes a raw state $s_t$ (and, in recurrent variants, a hidden state $h_{t-1}$) and outputs either a distribution over actions or an estimate of expected return. The "deep" component refers to the stacked layers that learn hierarchical features from high-dimensional inputs, enabling scalable solutions in complex environments. Training proceeds by sampling trajectories $\{s_t, a_t, r_t, s_{t+1}\}$ and updating $\theta$ to maximize the expected return

$$J(\pi_\theta) \;=\; \mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{T} \gamma^t\, r_t\Big].$$

Gradient estimators such as the policy gradient theorem allow backpropagation of $\nabla_\theta J$ through the network.

**Table 2.4:** Notation for Reinforcement Learning, MDP and PPO-related symbols

| Symbol | Meaning | Unit / Domain |
|---|---|---|
| $\mathcal{S}$ | Set of all states (e.g. battery SoC) | – |
| $\mathcal{A}$ | Set of all actions (e.g. charge power level) | – |
| $P(s' \mid s, a)$ | Transition probability | – |
| $R(s,a)$ | Immediate reward | currency / kWh |
| $\gamma$ | Discount factor | [0,1] |
| $\pi_\theta(a \mid s)$ | Parameterized policy | – |
| $J(\pi_\theta)$ | Expected return | cumulative reward |
| $V^\pi(s)$ | Value of state $s$ | cumulative reward |
| $Q^\pi(s,a)$ | Value of $(s,a)$ | cumulative reward |
| $\hat{A}_t$ | Advantage estimate at time $t$ | same as reward |
| $r_t(\theta)$ | Probability ratio: $\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$ | – |
| $\epsilon$ | Clipping parameter in PPO | – |
| $\mathcal{L}^{\text{CLIP}}(\theta)$ | PPO clipped objective | – |
| $\theta$ | Network parameters | – |
| $\alpha$ | Learning rate | – |
| $h_t$ | Recurrent hidden state | vector |

To ground these concepts, Figure 2.13 shows a simple gridworld in which an agent (e.g. a cleaning robot) must navigate to clean a "dirty" room and then return to a charging station. Each cell $S_i$ represents a distinct state at each time step the agent chooses one of four actions (up, down, left, right) and transitions according to the MDP dynamics. The reward function penalizes remaining dirty and incentivizes reaching the charging station before battery depletion. This toy environment illustrates how states, actions, transition probabilities, and rewards come together in a Markov decision process.

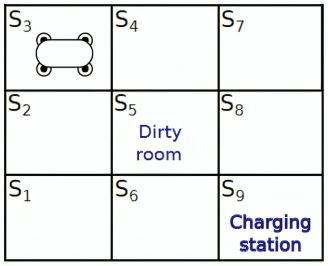



**Figure 2.13:** Gridworld MDP for a cleaning robot: the agent starts in one of the nine states $S_1 \dots S_9$, must visit the dirty room at $S_5$, and then navigate to the charging station at $S_9$. Transitions occur on up/down/left/right actions, and the reward structure encourages cleaning and timely return to charge.

#### 2.3.2.1 Markov Decision Process (MDP)

A Markod descision process is a framework for modeling sequential decision-making under uncertainty. It captures the dynamics of an environment in terms of states, actions, transitions, and rewards, providing the foundation upon which RL algorithms optimize behavior. An RL problem is formalized as an MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$. At each time $t$, the agent in state $s_t \in \mathcal{S}$ selects action $a_t \in \mathcal{A}$ according to $\pi_\theta(a_t \mid s_t)$, transitions with probability $P(s_{t+1} \mid s_t, a_t)$, and receives reward $r_t = R(s_t, a_t)$. The goal is to find $\pi_\theta$ that maximizes the discounted return $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$.

#### 2.3.2.2 Proximal policy optimization & Recurrent PPO algorithms

Proximal Policy Optimization (PPO) updates the policy by maximizing the clipped surrogate objective seen in 2.10, where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ and $\hat{A}_t$ is an advantage estimate (e.g. GAE). This constraint on $r_t$ prevents excessively large policy updates.

Recurrent PPO (RPPO) extends PPO by embedding an RNN (LSTM or GRU) to maintain a hidden state $h_t$, so that $\pi_\theta(a_t \mid s_t, h_{t-1})$ can condition on past observations. During training, trajectories carry both $s_t$ and $h_{t-1}$, and gradients propagate through time. This enables the agent to handle partial observability and learn temporal abstractions in environments with latent dynamics.

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\Big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\Big)\Big] \tag{2.10}$$

## 2.4 Performance metrics

As shown in Table 2.5, a suite of standard forecasting performance metrics is employed to evaluate model accuracy and robustness. These metrics quantify errors between predicted values $\hat{y}_t$ and actual observations $y_t$ over a validation set of size $N$.

By combining absolute, squared, percentage, and robust loss functions, the evaluation framework captures different aspects of error behavior ranging from average deviations to sensitivity toward large outliers. Together, these metrics provide a comprehensive understanding of model performance, enabling comparison across varying error scales and distributions.

The first group of metrics Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) focuses on error magnitude. MAE computes the average of absolute differences $|\hat{y}_t - y_t|$, treating all deviations equally and offering robustness to extreme values. In contrast, MSE squares each error term $(\hat{y}_t - y_t)^2$, thereby penalizing larger errors more heavily and emphasizing outliers. RMSE, defined as the square root of MSE, restores the error units to match those of the original data, making interpretation more intuitive while still retaining the squared-error emphasis on larger deviations.

The remaining metrics capture relative and distributional aspects of error. Mean Absolute Percentage Error (MAPE) expresses the average absolute error relative to the true value $y_t$, facilitating comparisons across series with different scales but suffering undefined values when $y_t = 0$.

Symmetric Mean Absolute Percentage Error (sMAPE) addresses this limitation by symmetrizing the denominator as $\Big(|y_t| + |\hat{y}_t|\Big)/2$, thus bounding the measure between 0% and 200% and reducing extreme percentage errors for values near zero.

The coefficient of determination $R^2$ quantifies the proportion of variance in $y_t$ explained by the model, ranging from $-\infty$ to 1, with values closer to one indicating a better fit.

Finally, Huber Loss combines the properties of MAE and MSE by using a quadratic penalty for errors within a threshold $\delta$ and a linear penalty for larger residuals, offering a balanced approach that is less sensitive to outliers than MSE but more discriminative than MAE.

**Table 2.5:** Forecasting Metrics Used in This Project

| Metric | Formula | Description |
|---|---|---|
| MAE | $\frac{1}{N}\sum_{t=1}^{N}\lvert\hat{y}_t - y_t\rvert$ | **Mean Absolute Error:** Average of the absolute differences between predicted values $\hat{y}_t$ and true values $y_t$. Treats all errors equally and is robust to outliers. |
| MSE | $\frac{1}{N}\sum_{t=1}^{N}(\hat{y}_t - y_t)^2$ | **Mean Squared Error:** Average of the squared differences between $\hat{y}_t$ and $y_t$. Penalizes larger errors more heavily, making it sensitive to outliers. |
| RMSE | $\sqrt{\frac{1}{N}\sum_{t=1}^{N}(\hat{y}_t - y_t)^2}$ | **Root Mean Squared Error:** Square root of MSE, providing an error measure in the same units as $y_t$. Emphasizes larger errors similarly to MSE. |
| MAPE | $\frac{100\%}{N}\sum_{t=1}^{N}\left\lvert\frac{\hat{y}_t - y_t}{y_t}\right\rvert$ | **Mean Absolute Percentage Error:** Error as a percentage of the true value $y_t$. Useful for relative accuracy but undefined when $y_t = 0$. |
| sMAPE | $\frac{100\%}{N}\sum_{t=1}^{N}\frac{\lvert\hat{y}_t - y_t\rvert}{\left(\lvert y_t\rvert + \lvert\hat{y}_t\rvert\right)/2}$ | **Symmetric Mean Absolute Percentage Error:** Bounded between 0% and 200%, mitigates extreme percentage errors when values are near zero by symmetrizing numerator and denominator. |
| $R^2$ | $1 - \frac{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{N}(y_t - \bar{y})^2}, \quad \bar{y} = \frac{1}{N}\sum_{t=1}^{N} y_t$ | **Coefficient of Determination** Proportion of variance in $y_t$ explained by the model. Ranges from $-\infty$ to 1; values closer to 1 indicate better fit. |
| Huber Loss | $\begin{cases} \frac{1}{2}(y_t - \hat{y}_t)^2, & \lvert y_t - \hat{y}_t\rvert \le \delta, \\ \delta\left(\lvert y_t - \hat{y}_t\rvert - \frac{1}{2}\delta\right), & \text{otherwise,} \end{cases}$ | **Huber Loss:** Combination of MSE and MAE that is quadratic for small errors ($\lvert y_t - \hat{y}_t\rvert \le \delta$) and linear for large errors ($\lvert y_t - \hat{y}_t\rvert > \delta$). Balances sensitivity to outliers; $\delta$ is the threshold parameter. |

# 3

# Methods

This chapter explains how the Home Energy AI system was built, from raw data to live control. It documents the full data acquisition, modelling, control design, automation, and validation. The study begin with a bird's-eye workflow, before detailing the two main components of the pipeline:

(i) multi-modal forecasting of demand, price and solar production and,

(ii) a safety-constrained reinforcement-learning agent that decides on battery usage.

Each subsequent section focuses on one link in that chain, describing the rationale behind key design choices, the data and tools used, and the procedures applied to verify performance.

Throughout, emphasis is placed on reproducibility, robustness, and relevance to Swedish residential tariffs and climate data. Prefect as the orchestration engine was chosen because of its reliability, ease of scheduling, failure handling and great documentation.

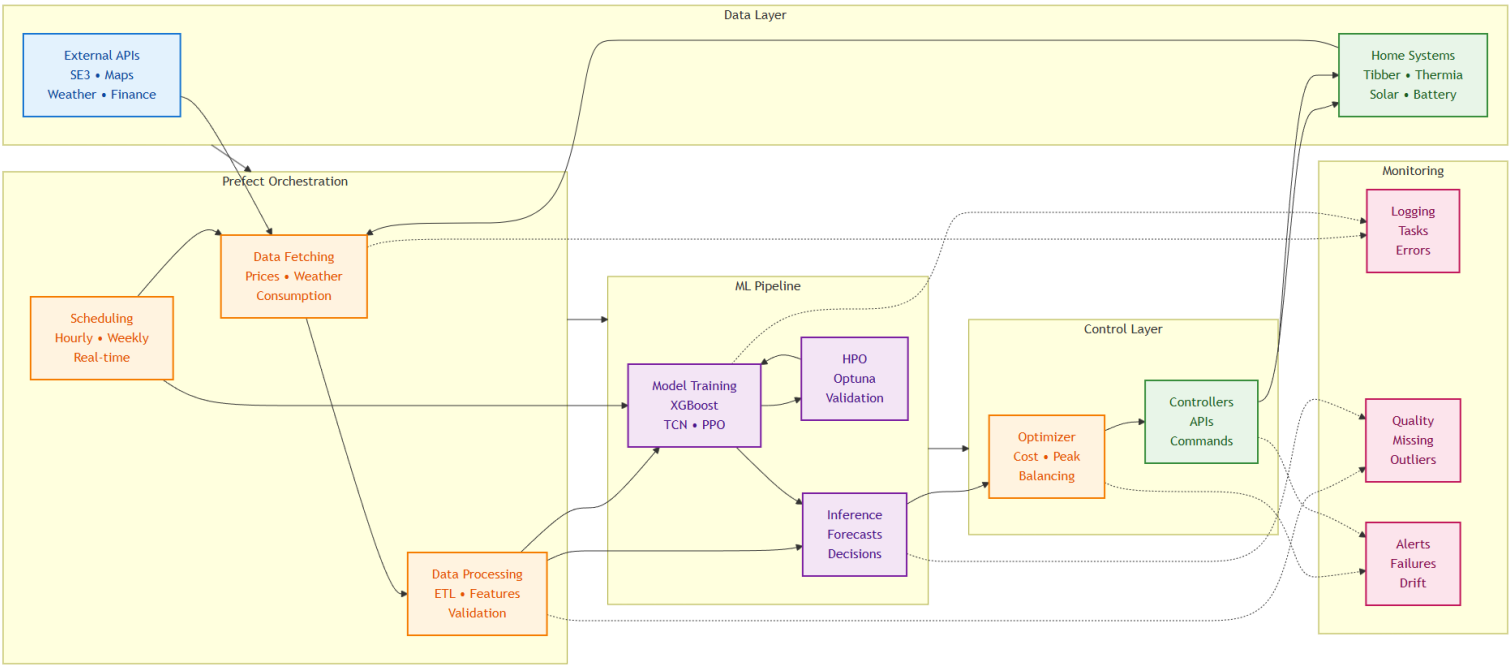



**Figure 3.1:** System architecture

## 3.1 Work-flow overview

Figure 3.1 sketches the end-to-end data–model–control workflow that turns raw measurements into real-time commands for the house. The diagram is organized in lanes corresponding to each logical layer of the software stack.

**Data layer**
Flows, implemented in Prefect[1], run at weekly, hourly and 15-minute intervals to ingest external data (spot prices, weather fields, commodity indices) and internal measurements (household consumption, heat-pump metrics, solar output). Each flow's schedule is defined in code so that timing is predictable and failures automatically retry with logs. Raw API responses are passed through an ETL (Extract, Transform, Load) process, timestamps are aligned, basic validity checks (gaps, outliers) are applied, and lagged/rolling features are computed. Any detected anomalies are sent to the monitoring layer for validation. (See Section 3.2 for details.)

**Machine learning pipeline**
To prevent concept drift, all forecasting models and the RL agent are retrained on a weekly basis. The forecasting stack and the reinforcement-learning agent each have dedicated modules (Sections 3.3 and 3.4, respectively). In brief, weekly flows assemble the latest cleaned data, run hyperparameter optimization for each forecasting model, and update model artifacts so that the control layer always uses up-to-date predictions.

**Control layer**
Every 15 minutes, a recurrent PPO agent (Section 2.3.2.2) evaluates the current state (prices, forecasts, state-of-charge, capacity-fee context) and proposes a battery control action. That action is filtered by a deterministic safety module, enforcing charge/discharge limits, SoC bounds and breaker constraints, before any command is sent to the house APIs. (Further orchestration details appear in Section 3.5.)

[1] Prefect is an open-source workflow orchestration tool [16]

## 3.2 Data acquisition & pre-processing

All raw signals enter through two pipelines.

**15-Minute Home Data Update**
Executes every 15 minutes. It gathers home-level measurements needed by the control agent: energy consumption, battery state-of-charge, heat-pump metrics and recent weather forecasts. All readings are aligned to Europe/Stockholm time to ensure consistency.

**Hourly Exogenous Data Update**
Executes every 15 minutes. This retrieves external inputs, day-ahead spot prices, $CO_2$ intensity, fuel costs, grid-mix breakdown and coarse weather forecasts, then aligns them to CET and forward-fills missing values for a continuous hourly series.

Both pipelines use modular scripts that apply basic integrity checks (timestamp, gap detection and numerical validation) before feature engineering. The cleaned, feature-rich tables feed directly into the machine learning and control pipelines.

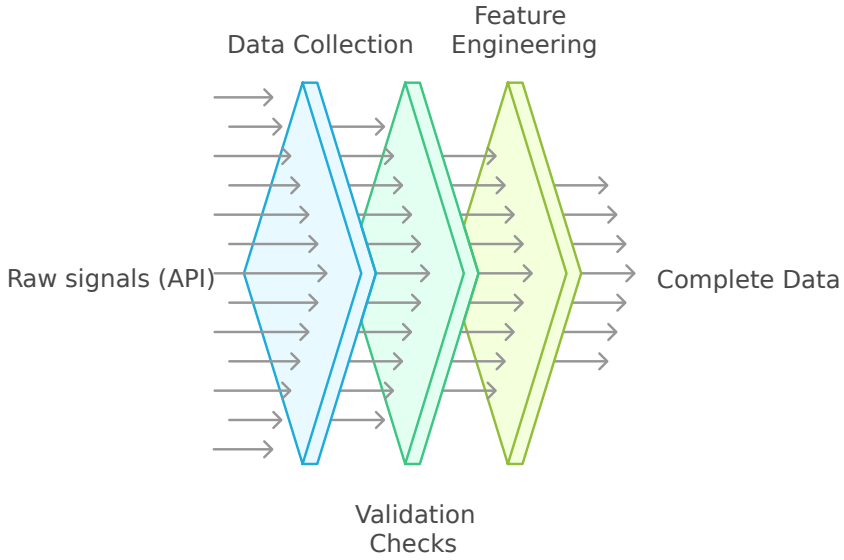



**Figure 3.2:** Data acquisition pipeline showcasing the raw API signal process through validation and feature engineering to complete usable and clean data for the ML stack

## 3.3 Forecasting models

Effective forecasting is at the heart of our Home Energy AI system where accurate price, demand, and solar predictions feed directly into the control agent's decisions. In this chapter, we describe three forecasting streams, electricity price (Section 3.3.1), home demand (Section 3.3.2), and solar production (Section 3.3.3) each built and validated separately before being stitched together for real-time operation.

We start by explaining why we opted for a trio of specialized sub-models rather than a single predictor (Trend, Peak, Valley for prices), and how we merge their outputs into a coherent hourly price curve. Next, we explain the demand model that captures household consumption patterns at a 15-minute resolution, followed by the solar model that uses weather and array geometry data to forecast PV output. Each section covers data inputs, feature engineering, model architecture, training regimen, and performance metrics. Our goal isn't to chase perfect accuracy but to provide "good enough" forecasts, refreshed weekly, so that the RL agent can optimize battery charge/discharge decisions with minimal latency and maximal robustness.

### 3.3.1 Price Model

Forecasting electricity prices is tough, prices depend on many factors (supply, demand, weather, fuel costs, transmission) [4].

Europe is calculating day ahead prices by using the EUPHEMIA market-coupling algorithm across zones [17]. Even SE3 prices, our target, reflect cross-border flows via that same process, making the prediction system difficult [18].

Building one "all-in-one" model with every feature can capture subtle patterns, but it demands hard-to-get data, huge compute, and adds complexity [4].

Instead, we use three focused sub-models:

- "Trend" for overall price level.
- "Peak" detector for high-price hours.
- "Valley" detector for low-price hours.

Because merging multiple models can amplify forecast errors [4], we adopt the "specialized + merge" strategy where each smaller model is faster to train and can be updated weekly with proper validation, yet still delivers predictions for battery control.

#### 3.3.1.1 Trend Model

| Category | Features |
|---|---|
| **Grid** | fossilFreePercentage, renewablePercentage, powerConsumptionTotal, powerProductionTotal, powerImportTotal, powerExportTotal, nuclear, wind, hydro, solar, unknown, import_SE-SE2, export_SE-SE4, import_NO-NO1, export_NO-NO1, import_DK-DK1, export_DK-DK1, import_FI, export_FI |
| **Prices** | PriceArea, SE3_price_ore, price_24h_avg, price_168h_avg, price_24h_std, hour_avg_price, price_vs_hour_avg, Gas_Price, Coal_Price, CO2_Price |
| **Weather** | temperature_2m, cloud_cover, relative_humidity_2m, wind_speed_100m, wind_direction_100m, shortwave_radiation_sum |
| **Time & Holiday** | hour_sin, hour_cos, day_of_week_sin, day_of_week_cos, month_sin, month_cos, is_morning_peak, is_evening_peak, is_weekend, season, is_holiday, is_holiday_eve, days_to_next_holiday, days_from_last_holiday |

**Table 3.1:** All input features by category. Data range starts at 2017-01-01

The Trend Model uses XGBoost to forecast the next 24 hours of SE3 day-ahead spot prices (öre/kWh) at an hourly resolution. Its main purpose is to capture the smooth backbone of the price curve so that subsequent models can focus on extreme spikes and dips.

**Features.**
All features listed in Table 3.1 are generated by the hourly data acquisition pipeline (Section 3.2). These include grid-related metrics, historical price statistics, weather variables, and time/holiday encodings. Each hourly record represents a snapshot of the system state, with appropriate lagged and rolling statistics already applied.

**Hyperparameter tuning.**
Every week, an Optuna study samples trials to identify optimal hyperparameters. The objective is squared-error regression, using RMSE as the evaluation metric, with the "hist" tree method and "depthwise" growth policy. The best-performing model on a rolling hold-out validation set is tagged as the current production version.

**Training & validation.**
Training occurs weekly on the most recent 3–6 months of hourly data. We employ time-series cross-validation with forward-rolling splits, holding out one day at a time. Models that improve validation metrics replace the existing production model.

**Inference.**
At runtime, the Trend Model outputs a 24-element vector of hourly price forecasts. These baseline predictions are later combined with outputs from Peak and Valley detectors (Section **??**) to form a composite forecast that balances overall trend accuracy with sensitivity to extreme price events.

#### 3.3.1.2 Peak & Valley Models

Both the Peak and Valley Models use a temporal-convolutional network (TCN) to classify each of the next 24 hours as an extreme event (peak or valley) or not. Instead of predicting a continuous price, each model outputs a 24-element binary vector, where "1" denotes a predicted peak (or valley) hour.

**Features.**
Both models reuse the feature set in Table 3.1 (Section 3.3.1.1). Inputs are constructed as a sliding window of the past 168 hours (one week) of these features, yielding a 168 $\times$ (feature-count) tensor for each model.

**Peak labeling.**
Before training the models, we first needed to define what constitutes a "peak" or "valley" in the price series, a seemingly easy task on its own. To do this we scanned historical prices and an hour is marked as a peak if it passes a derivative check that capture sharp rise-then-drop points. These labels (see Figure 4.1) serve as ground truth for Peak Model training.

**Valley labeling.**
By inverting the peak criteria, an hour is labeled a valley if it passes a negative derivative check, thus capturing sharp drop-then-rise patterns. These binary valley labels form the training targets for the Valley Model (see Figure 4.2).

**Architecture.**
Each TCN has two stacked residual blocks:

- Three 1D convolutions (kernel 3, dilations 1, 2, 4), 64 filters each, followed by BatchNorm, ReLU, dropout, and a skip connection.
- Three 1D convolutions (kernel 3, dilations 8, 16, 32), same pattern.

A global average layer collapses the temporal dimension, and a final dense layer with 24 sigmoid outputs produces per-hour probabilities. Both models share this structure; they differ in class-weight and loss settings.

**Training & Validation.**
Because peaks and valleys account for $< 5\%$ of hours, we apply SMOTE[2] to oversample minority class examples, increasing extreme event labels to $\approx$20% of training data. After training, we sweep the probability threshold to balance precision and recall.

**Inference & Performance.**
At inference, a threshold converts probabilities into binary flags. We visualize the model's output by mapping predicted probabilities to marker heights, so higher probabilities indicate a stronger predicted extreme (see Figure 4.3(a&b)). However, this approach is flawed, since the probabilities do not directly correspond to actual peak or valley magnitudes, it was chosen purely for ease of implementation.

[2]SMOTE (Synthetic Minority Oversampling Technique) is a machine learning technique used to address class imbalance in datasets. It generates synthetic samples of the minority class to help balance the class distribution and improve model performance.

#### 3.3.1.3 Merged Model

After training the Trend, Peak, and Valley Models independently, we merge their outputs into a single prediction. It does this with a simple override logic, where any hour flagged as a peak or valley replaces the Trend Model's forecast for that hour multiplied by the labeled prediction probability to make the price magnitude. Hours not flagged by either extreme model keep the Trend Model's value, see Figure 4.9 for reference. This is a suboptimal merging sinc the price extreme models is not directly trained to predict the magnitude of an hour, this merging solution was used for its simplicity.

### 3.3.2 Home Demand Model

The Home Demand Model forecasts hourly residential electricity consumption using an XGBoost regressor with extensive feature engineering. Rather than classifying extreme events, this model predicts a continuous consumption value (kWh) for each of the next 24 hours, enabling the downstream controller to schedule battery usage optimally.

**Features.**
We engineer over 220 predictors drawn from historical consumption, calendar signals, weather data, and home-specific dynamics. Many of these are lagged features (e.g., consumption at 1 h, 2 h, 3 h, 24 h, 48 h, 72 h, and 168 h ago, plus rolling statistics over 6 h, 12 h, 24 h, 48 h, 72 h, and 168 h windows) to capture temporal autocorrelation. For the full list of non-lagged features (time encodings, temperature, wind speed, solar irradiance, etc.), see Table 3.1.

During this project, there were no available data for usage occupancy patterns, which is a highly correlated feature for correctly predicting home energy demand. To address this, we fit a three-state Gaussian Hidden Markov Model (HMM) on historical hourly load data. Each hour's observed consumption is treated as an emission, and the HMM learns three latent "occupancy" states roughly corresponding to *low*, *medium*, and *high* home activity. At each time step, we compute the posterior probability of each HMM state given the past sequence of loads. These three posterior probabilities are included as features, allowing the model to infer whether occupants are likely away (state 1), partially present (state 2), or fully present (state 3). By providing this latent occupancy signal, the XGBoost regressor can adjust its forecast when people are more or less active in the home.

The only available data regarding the power-hungry heat pump is its binary state (running or not). We calculate estimated consumption explicitly via two intermediate calculations:

**Thermal Output Calculation**:

$$\dot{Q}_{\text{heat}} \;=\; \dot{m} \times c_p \times \Delta T_{\text{clamped}},$$

where
$\Delta T_{\text{clamped}} = \min\Big(\max(\Delta T,\ 2\ \text{K}),\ 10\ \text{K}\Big)$
is the supply–return water temperature difference clamped between 2 K and 10 K.

**Electrical Input Estimation**:

$$P_{\text{in}} = \frac{\dot{Q}_{\text{heat}}}{\text{COP}},$$

using a fixed Coefficient of Performance (COP). Including both $\dot{Q}_{\text{heat}}$ and $P_{\text{in}}$ as features captures the nonlinear relationship between ambient/indoor temperatures and heat pump electrical load.

**Architecture.**
We employ an XGBoost regressor in `reg:squarederror` mode to learn nonlinear mappings from the 220+ features to hourly consumption. Optuna is used to tune the following hyperparameters:

- `n_estimators`: 200 – 1500
- `max_depth`: 4 – 12
- `learning_rate`: 0.005 – 0.2 (log-scaled)
- `subsample`: 0.7 – 0.95
- `colsample_bytree`: 0.7 – 0.95
- `reg_alpha`: $10^{-8}$ – 10 (log-scaled)
- `reg_lambda`: $10^{-8}$ – 10 (log-scaled)
- `min_child_weight`: 1 – 7
- `gamma`: 0 – 0.5

These hyperparameters are searched over 500 Optuna trials, using time-series cross-validation on the training window.

**Inference & Performance.**
During inference, the model generates point predictions for each of the next 24 hours. Prediction uncertainty is estimated via the variance of leaf outputs in the trained trees. Model validation is performed on a hold-out dataset to assess generalization before deployment.

### 3.3.3 Solar Prediction Model

To estimate the photovoltaic (PV) energy output for our installation, we relied on the forecast.solar[3] API rather than developing a custom model. This service handles the necessary physical and meteorological computations (e.g., irradiance decomposition, angle of incidence adjustments) described in the Theory chapter, allowing us to obtain reliable hourly forecasts spanning multiple days.

Our PV system comprises two arrays oriented at different azimuths (see Figure 3.3):

- Southeast-facing array: 24 panels with a tilt of 30.
- Northwest-facing array: 26 panels also tilted at 30.

We issue separate API calls to forecast.solar for each orientation to account for differences in irradiance and shading throughout the day. After receiving the two hourly time series, one for the southeast array and one for the northwest array, we sum their outputs at each hour to obtain the total predicted energy production.

To align the forecasts with our panels' real-world characteristics (manufacturer tolerances, inverter efficiency, wiring losses, etc.), we apply a small scaling factor to each orientation's forecast. This factor was determined empirically based on prior operational data by comparing historical production data (hourly aggregated) against the API's predictions over a representative validation period. The resulting forecast can be visualized in Figure 4.6.

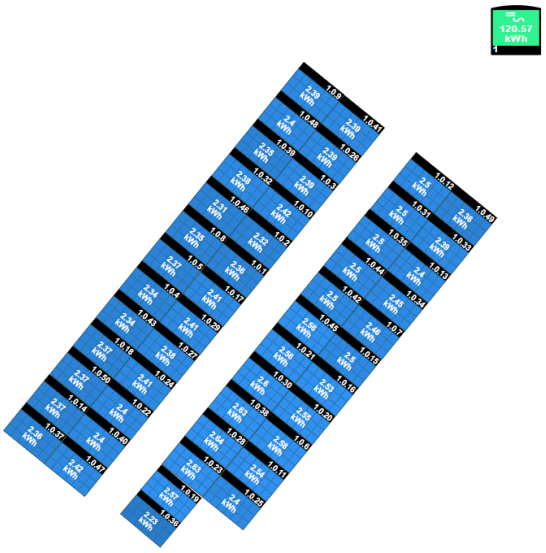



**Figure 3.3:** Layout of the dual-orientation PV array with per-panel energy output (in kWh), where North is up

[3]forecast.solar is a cloud-based forecasting service that leverages PVGIS (Photovoltaic Geographical Information System) for irradiance and weather-based solar production estimates via a simple RESTful API [19]

## 3.4 Reinforcement learning agent

The core of our control strategy is a reinforcement learning (RL) agent designed to optimally manage battery charge and discharge in response to predicted electricity prices, forecasted solar production, and household load. Instead of using a rule-based or optimization-only approach, we implement a recurrent Proximal Policy Optimization (RPPO) agent with LSTM memory to capture temporal dependencies in price, solar, and load patterns. By interacting with a custom Gymnasium environment[4], the agent learns a policy that balances cost minimization (including spot-price energy costs and monthly capacity fees) with battery state-of-charge (SoC) constraints.

### 3.4.1 Environment Design

The environment's observation (state) is a dictionary comprising the following components:

- **Battery SoC**: a continuous scalar in [0, 1] representing the fraction of the battery's usable capacity currently stored.
- **Time Index**: a 3-element vector: hour of day, minute of hour, day of week to encode patterns.
- **Price Forecast**: a 24-hour ahead vector of predicted spot prices (SE3 market), in öre/kWh, updated daily.
- **Solar Forecast**: a 72-hour ahead vector of predicted PV production (kW), obtained from the forecast.solar API.
- **Capacity Metrics**: a 5-dimensional vector top1, top2, top3, rolling average, month progress that tracks the three highest grid import peaks encountered in the current billing month (for capacity-fee calculation), the rolling average import, and how far through the month we are.
- **Price Averages**: two scalars 24 hourly average, 168 hourly average representing recent average spot prices, used to identify high- and low-price regimes.
- **Night Discount Flag**: a boolean (0 or 1) indicating whether the current hour falls into a low-tariff "night" period.
- **Load Forecast**: a 72-hour ahead vector of predicted household consumption (kW), produced by the demand forecasting model.

[4]Gymnasium is an open-source toolkit and framework for developing RL algorithms [**?**].

### 3.4.2 Reward Function

We design a multi-component reward at each hour $t$ to guide the agent toward cost-minimization while respecting SoC constraints and preferring to keep SoC in a comfortable mid-range. The total reward $r_t$ is the sum of:

1. **Grid Cost Penalty:** When net grid power $P_{g,t}$ (kW) is drawn from the grid, the instantaneous cost in öre is computed from the sum term in Equation (2.1) covered in Section 2.1. This penalizes importing power during expensive hours.

2. **Capacity Fee Penalty:** In Sweden's tariff structure, only the single highest grid import peak per day contributes to the monthly capacity fee. We maintain a list of top-3 peaks $\{(t_i, P_{\text{peak},i})\}$ in the current month. Whenever a new $P_{g,t}$ exceeds that day's recorded peak, we update accordingly. At the end of each month, the highest recorded peak (kW) is multiplied by the constant tariff (öre/kW).

3. **SoC Penalty:** We encourage the agent to keep the battery's SoC within a preferred range $[\text{SoC}_{\text{low}}, \text{SoC}_{\text{high}}]$. The complete SoC reward is shown in Equation (3.1). Where, $\alpha$ is a large penalty factor for violating hard limits $[\text{SoC}_{\text{min}}, \text{SoC}_{\text{max}}]$, while $\beta$ softly rewards or penalizes departures from the preferred mid-range.

4. **Potential-Based Shaping:** To accelerate learning without altering the optimal policy, we add a potential-based shaping term [7]. Let $\Phi(\text{soc}_t)$ be a smooth potential function maximizing at $(0.3 + 0.8)/2 = 0.55$. Then at each step:

   $$r_{\text{shape},t} = \gamma\,\Phi(\text{soc}_{t+1}) \;-\; \Phi(\text{soc}_t),$$

   where $\gamma = 0.99$ is the PPO discount factor. This term (weight $w_{\text{shape}}$) gently guides the agent's SoC toward the "optimal" mid-range over time, speeding up convergence.

5. **Battery Degradation Penalty:** Cycling the battery (charging or discharging) results in wear costs. We penalize absolute energy throughput $|E_{\text{throughput},t}|$ (in kWh) by

   $$w_{\text{deg}} \times \left|E_{\text{throughput},t}\right| \times 45\,\text{öre/kWh},$$

   45 öre/kWh here is calculated by the initial cost $\frac{C_{init}}{(N_{cycles} \times E_{capacity})}$, which is the actual cost per every kWh of usage [10]. This term is symmetric for charge vs. discharge and discourages excessive cycling.

6. **Action Modification Penalty:** When the agent proposes an action $a_t$ that violates safety constraints (e.g., would drive $\text{soc}_{t+1}$ outside $[SoC_{min}, SoC_{max}]$), the environment's safety mask overrides it. Each override incurs a penalty. Consecutive violations escalate the multiplier. This discourages unsafe exploration and encourages the agent to learn feasible actions.

7. **Arbitrage Bonus:** This is the core profit mechanism. We define dynamic thresholds based on recent price percentiles:

   $P_{\text{low}} = 30\text{th}$ percentile of last 24 h,

   $P_{\text{high}} = 75\text{th}$ percentile of last 168 h.

The agent then gets the reward by charging when $Price < P_{low}$ and discharging when $Price > P_{high}$

8. **Export Bonus:** When the agent discharges into the grid, we also reward exported energy $E_{\text{exp},t}$ (kWh) by:

   $$w_{\text{export}} \times \Big(\text{spot}_t + tax_{bonus}\Big) \times E_{\text{exp},t},$$

   Adding a $tax_{bonus}$ of 60 öre/kWh simulates the current swedish tax reduction on exported energy, so discharging during moderately high prices yields extra incentive. This encourages the agent to export when it is both profitable and grid-friendly.

9. **Night Charging Reward:** To further incentivize off-peak charging, any energy $E_{\text{night},t}$ charged during the "night" window (22:00–06:00) receives a bonus

10. **Solar-Aware SoC Management:** Make room for solar charging when significant production is expected. Logic: At each time step, look ahead 6–12 hours using the solar forecast. If the forecasted incoming solar energy exceeds 2.0 kWh and the current state of charge $\text{soc}_t > 0.60$, assess available battery headroom $H_t = 1 - \text{soc}_t$. Calculation: If $H_t < 0.50 \times E_{\text{solar\_6–12h}}$, then any discharge $E_{\text{dis},t}$ that creates additional headroom is rewarded. In other words, for every kWh discharged in anticipation of soon available solar production, the agent gains a bonus.

11. **Night-to-Peak Chain Bonus:** Encourage a chain strategy of charging at night when prices are low and discharging during subsequent peak hours. It tracks $E_{\text{night\_charged}}$, the energy (kWh) drawn from the grid between 22:00 and 06:00 each day. This summed pool decays after 24 hours (i.e., energy charged more than 24 h ago is discarded). During any "peak" hour (defined as either $Price > P_{\text{high}}$ or observed household load $> L_{\text{high}}$), any discharge $E_{\text{chain},t}$ that can be matched against the current pool is rewarded.

    In practice, if the agent discharged 1 kWh during a peak hour that was originally charged at night (within the last 24 h), it receives a bonus.

$$\text{Reward}(\text{soc}_t) = \begin{cases} -\alpha \times \Big(1 + \text{severity}\Big), & \text{soc}_t \le \text{SoC}_{\text{min}} \text{ or } \text{soc}_t \ge \text{SoC}_{\text{max}}, \\ +\beta \times \left(1 - \frac{\left|\text{soc}_t - (\text{SoC}_{\text{low}} + \text{SoC}_{\text{high}})/2\right|}{(\text{SoC}_{\text{high}} - \text{SoC}_{\text{low}})/2}\right), & \text{SoC}_{\text{low}} \le \text{soc}_t \le \text{SoC}_{\text{high}}, \\ -\beta \times \frac{\text{SoC}_{\text{low}} - \text{soc}_t}{\text{SoC}_{\text{low}} - \text{SoC}_{\text{min}}}, & \text{soc}_t < \text{SoC}_{\text{low}}, \\ -\beta \times \frac{\text{soc}_t - \text{SoC}_{\text{high}}}{\text{SoC}_{\text{max}} - \text{SoC}_{\text{high}}}, & \text{soc}_t > \text{SoC}_{\text{high}}. \end{cases} \tag{3.1}$$

The *net* reward at time $t$ is then shown in Equation (3.2) or in the more compact form here:

$$R(t) \;=\; \mathbf{w}^{\mathsf{T}}\,\mathbf{R}(t) \;=\; \sum_{i=1}^{N} w_i\, R_i(t),$$

where we collect all reward components $R_i$ into a vector $\mathbf{R}(t) \in \mathbb{R}^N$ and all corresponding weights $w_i$ into $\mathbf{w} \in \mathbb{R}^N$.

$$\mathbf{R}(t) \;=\; \begin{bmatrix} R_1(t) \\ R_2(t) \\ R_3(t) \\ R_4(t) \\ R_5(t) \\ R_6(t) \\ R_7(t) \\ R_8(t) \\ R_9(t) \\ R_{10}(t) \\ R_{11}(t) \end{bmatrix} \;=\; \begin{bmatrix} -\,\text{grid\_cost}(t) \\ -\,\text{capacity\_penalty}(t) \\ -\,\text{degradation\_cost}(t) \\ \text{soc\_reward}(t) \\ \text{shaping\_reward}(t) \\ \text{night\_charging}(t) \\ \text{arbitrage\_bonus}(t) \\ \text{export\_bonus}(t) \\ -\,\text{action\_penalty}(t) \\ \text{solar\_soc\_reward}(t) \\ \text{night\_peak\_chain}(t) \end{bmatrix}, \quad \mathbf{w} \;=\; \begin{bmatrix} w_{\text{grid}} \\ w_{\text{cap}} \\ w_{\text{deg}} \\ w_{\text{soc}} \\ w_{\text{shape}} \\ w_{\text{night}} \\ w_{\text{arbitrage}} \\ w_{\text{export}} \\ w_{\text{action\_mod}} \\ w_{\text{solar}} \\ w_{\text{chain}} \end{bmatrix}. \tag{3.2}$$

This reward function balances key objectives in residential energy storage: reducing electricity costs, preserving battery life, ensuring safety, and capturing revenue opportunities.

By splitting it into weighted components, we can tune the agent to prioritize cost savings during expensive periods and strategically store energy for future arbitrage.

The potential-based shaping and graduated penalties promote stable learning without altering the optimal policy, while solar-aware SoC management and night-to-peak chaining guide the agent to anticipate renewable generation and predictable load cycles.

Overall, this design lets the RL agent learn advanced, market and weather aware strategies that should outperform simple peak-shaving or time-of-use rules, achieving positive economic results while respecting operational limits.

### 3.4.3 Training Regime

We implement a sophisticated training regime incorporating several advanced techniques to ensure robust policy learning. The agent is trained using a three-phase **curriculum learning** schedule spanning millions of timestep:

Phase 1 (20% of training) focuses on basic battery management with simplified 3-day episodes, emphasizing SoC discipline with reduced reward complexity.

Phase 2 (30% of training) introduces price arbitrage strategies over 7-day episodes while maintaining basic constraints.

Phase 3 (50% of training) employs full system complexity with 30-day episodes, complete solar integration, and all reward components active.

**Adaptive exploration** is achieved through dynamic entropy coefficient scheduling, starting with high exploration (4× base entropy) in early training and gradually reducing to 0.5× base entropy in the final 20% of training to enable policy refinement.

**Continuous reward monitoring** analyzes component balance every 100,000 timesteps, detecting reward dominance, extreme value ranges, and providing automatic recommendations for hyperparameter adjustment.

**Data augmentation** introduces controlled variability during training through random scaling of solar production (±5%) and consumption patterns (±15%) to improve policy robustness across diverse operating conditions.

This training framework enables the agent to learn stable, interpretable policies that generalize effectively across varying seasonal conditions and market dynamics while maintaining strict adherence to operational safety requirements.

Table 3.2 summarizes the core hyperparameters of our RecurrentPPO agent.

**Table 3.2:** Trained Model Parameters

| **Parameter** | **Value** |
|---|---|
| Algorithm | RecurrentPPO |
| Learning Rate | $2 \times 10^{-4}$ |
| Discount Factor | 0.995 |
| Rollout Buffer | 4096 |
| Batch Size | 256 |
| Training Epochs | 8 |
| LSTM Layers | 1 |
| LSTM Hidden Size | 64 |
| Training Steps | 30,000,000+ |
| GAE Lambda | 0.95 |
| Entropy Coefficient | 0.01 |

### 3.4.4 Benchmark Strategies

To evaluate the RL agent's performance, we implement five rule-based battery management strategies representing conventional approaches used in residential energy storage systems.

**1. No Battery Baseline:** This strategy serves as the control case where no battery intervention occurs. The household directly imports or exports all net energy (consumption minus solar production) to the grid, establishing the worst-case scenario for capacity fees and energy costs.

**2. Time-of-Use Strategy:** A simple time-based approach using fixed scheduling rules:

- Night charging (22:00-06:00): Moderate charging to 60% SoC when solar production is minimal
- Evening discharge (16:00-20:00): Conservative discharge from 40% SoC during typical peak hours
- Power limits: 80% of maximum charging rate, 70% of maximum discharge rate

**3. Price-Based Strategy:** Uses fixed price thresholds based on typical Swedish electricity market ranges:

- Low-price threshold: 40 öre/kWh (conservative estimate)
- High-price threshold: 120 öre/kWh (conservative estimate)
- Conservative charging: 80% power rate when price  threshold
- Conservative discharge: 60% power rate when price  threshold

**4. Solar-Following Strategy:** Reacts only to current timestep solar conditions:

- Excess solar charging: Store 90% of excess production when solar $>$ demand
- Solar deficit discharge: Provide 80% of shortfall when solar $<$ 50% of demand
- Simple thresholds: 30-80% SoC operating range

**5. Peak-Shaving Strategy:** Uses simple historical peak tracking (24-hour memory) without demand forecasting:

- Dynamic threshold: Maximum of 5.0 kW or 80% of recent peak demand
- Peak reduction: Discharge when current demand exceeds threshold
- Excess storage: Charge with 80% of excess generation $>$ 1.5 kW

All strategies include battery degradation costs in their economic evaluation, calculated at 45 öre/kWh of energy throughput.

This comparison framework demonstrates the economic value of intelligent prediction and multi-objective optimization in complex energy management scenarios.

## 3.5 Orchestration & Automation

The orchestration is managed by a Python script using Prefect 3.4.1. Each basic operation, such as fetching data, training a model, or running an inference is defined as a Prefect task. Related tasks are grouped into flows that run on a schedule. All tasks use `run_python_script()` to ensure consistent retry behavior, timeouts, and logging. We assign descriptive names to tasks so the Prefect UI shows clear labels instead of file paths.

There are four main flows, each with its own schedule. An hourly flow updates external data (electricity prices, weather, $CO_2$ metrics, and solar forecasts). A 15-minute flow gathers home data (energy consumption, heat-pump status, and actual load) and then runs the reinforcement-learning (RL) agent to decide battery actions; this flow also generates a brief HTML/Markdown report with the agent's status. A daily flow runs each morning to refresh data and, if enabled, run price forecasts. Weekly flows retrain models, price models on Sundays, the demand model on Mondays, and the RL agent on Tuesdays (see Figure 3.4). These training jobs use adjustable hyperparameters and allow extra time to finish.

When the RL agent runs, we parse its output to extract battery SoC, energy commands, price signals, and action values. This information is included in a short report that flags whether the agent is charging, discharging, or idle, and summarizes solar and load forecasts. All logs, reports, and model files are saved as Prefect artifacts so any issues can be traced later (see Figure **??** in Section **??**).

Error handling is consistent across all flows: each task retries once after a brief delay if it fails, and any task that exceeds its timeout is stopped and logged. A single task's failure does not stop the entire flow, downstream steps either skip or use fallback logic. This design keeps data fetching, model training, and RL inference running smoothly with minimal manual intervention.

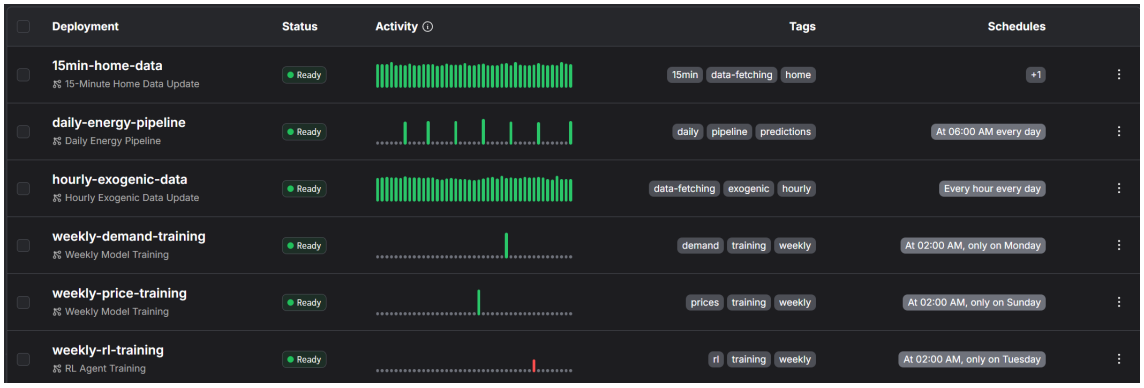



**Figure 3.4:** Prefect Dashboard showing the deployments for the scheduling of the Home system

## 3.6 Validation & Evaluation Procedures

To verify that each forecasting component and the RL controller perform as intended under realistic conditions, we employ a layered validation strategy. For the price and valley detectors where extreme events are sparse we emphasize balancing precision and recall via oversight on false positives/negatives.

Regarding the trend model, we rely on automated hyperparameter tuning with constrained Optuna trials, exploiting XGBoost's computational efficiency. Finally, the RL agent undergoes extensive testing on historical data. For each model, dedicated evaluation scripts generate performance plots using unseen test-split data, enabling manual inspection of any anomalous behavior because visual review often reveals issues that numerical metrics alone cannot fully capture.

The RL agent generates dedicated HTML/Markdown documentation of its current battery action decision shown in Figure 3.5 for logging and, more importantly, to provide visual feedback. This approach supports the project's goal of creating a highly user-friendly system that can be easily adapted in the future.

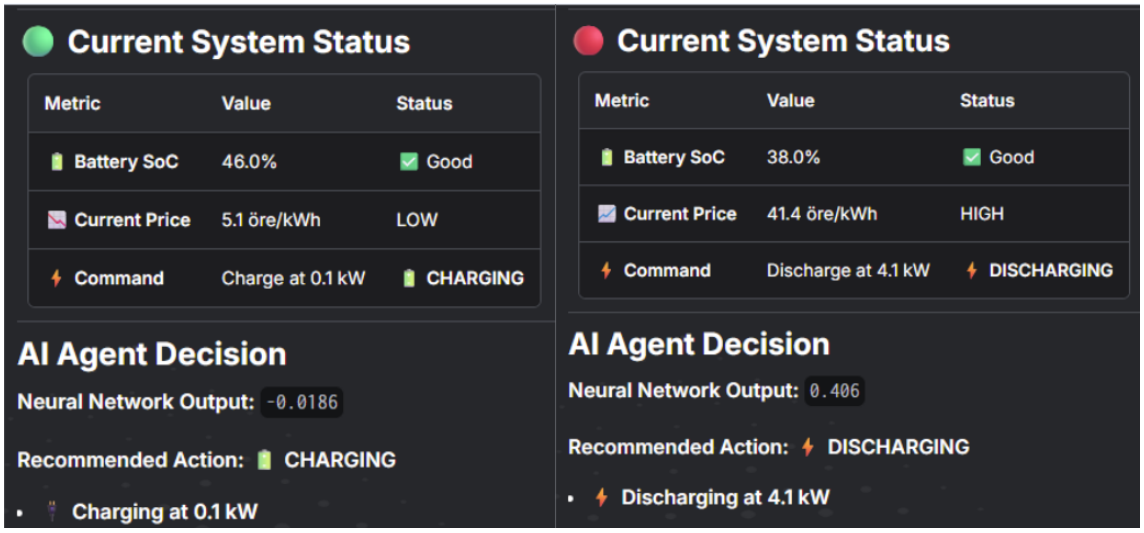



**Figure 3.5:** Prefect markdown artifact generated after running the agent before and after turning on the sauna, (left image being before and right after sauna has been on for some time.)

# 4
# Results

In this chapter, we present the performance outcomes for each component of the optimization platform on test data and in real-world deployment. We first report results for the price-extreme detectors (peaks/valleys) and the trend forecasting model, followed by solar-generation forecasts. Next, we summarize demand-forecast accuracy. Finally, we detail the RL agent's testing results.

## 4.1 Price Models

The following sections comprises three components: detecting extreme price events (peaks and valleys), forecasting the underlying price trend, and merging these outputs into a single, unified prediction. In the first subsection, we evaluate how accurately the peak and valley classifiers flag hours. Next, we assess the Trend Model's ability to capture broader hourly price movements over a month. Finally, the Merged Model section shows how combining the trend forecast with extreme-event flags improves overall prediction quality.

### 4.1.1 Price extreme detection

Figure 4.1 displays a two-week period, with ground-truth peaks marked by red triangles on the hourly spot price curve (blue). Similarly, Figure 4.2 highlights true valleys. These plots illustrate that extreme price events are quite rare, fewer than 5% of hours qualify as high-price peaks, where the low-price labels make up $\approx 20\%$ of the data.

Because these peak and valley labels are generated by a rule-based algorithm and treated as "hard truths," the resulting classifiers can still exhibit numerous false negatives and false positives. For example, when a peak persists for more than one hour, the algorithm may label only a single hour as a peak, even though we would ideally mark every hour of sustained high prices.

The same challenge applies to valley periods, despite extensive effort to ensure the labeling algorithm captures multiple consecutive low-price hours, it often fails to do so. Consequently, relying solely on summary metrics can be misleading. Visual inspection of these labeled events is therefore essential.

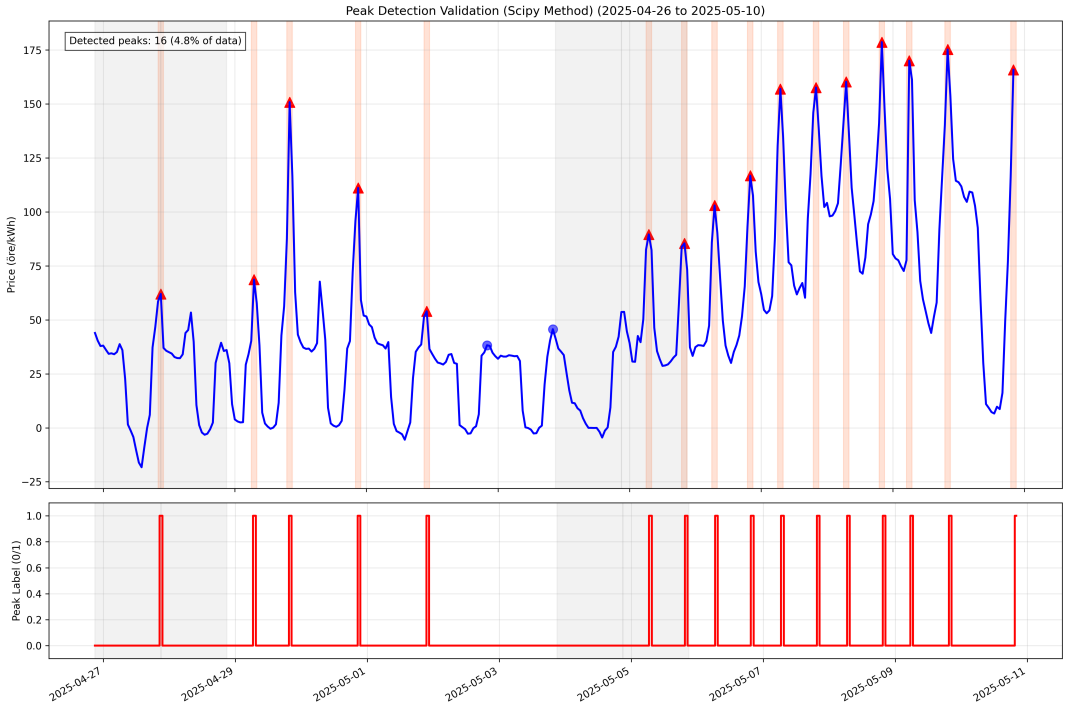



**Figure 4.1:** Hourly spot price (blue) with red triangles marking detected peaks (ground-truth) over a two-week window.

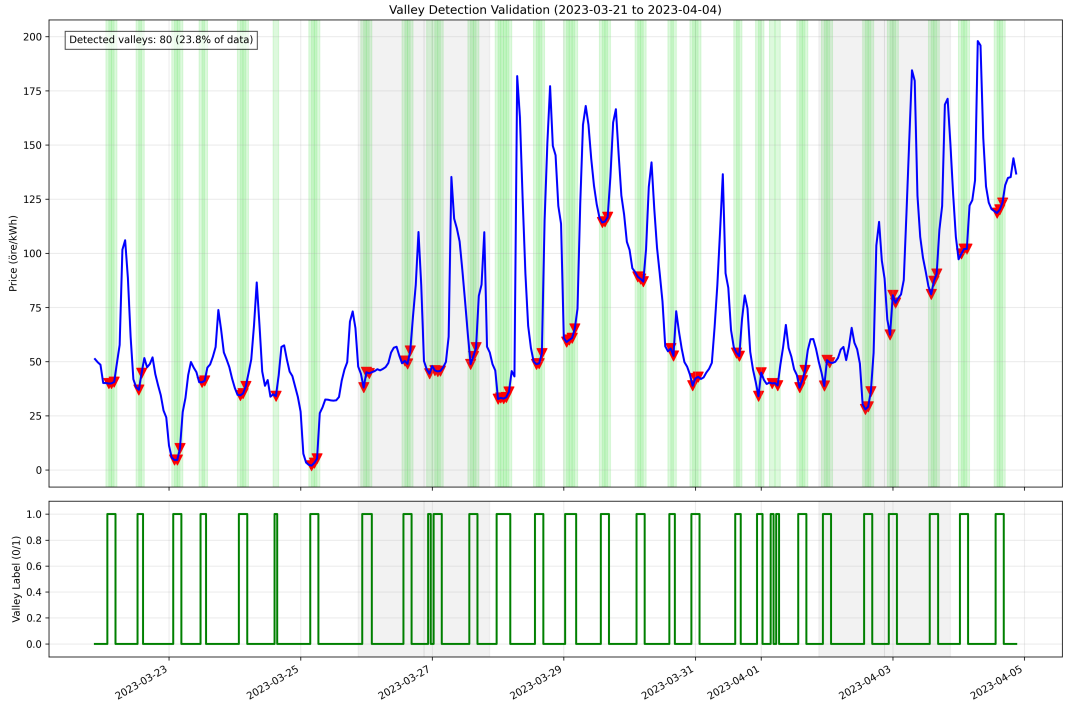



**Figure 4.2:** Hourly spot price (blue) with red triangles marking valleys

Figure 4.3a and 4.3b show the model performance over one week of labeled data for peaks and valleys, respectively.

**Peaks**
In Figure 4.3a, the blue line represents the hourly spot price, red triangles mark actual (ground-truth) peaks, and inverted green triangles mark predicted peaks. The shaded vertical bands indicate actual peaks. Over this week, there were 8 actual peaks, but the model predicted 16. This yields a precision of 0.44 but a recall of 0.88 (the model still catches roughly half of true peaks).

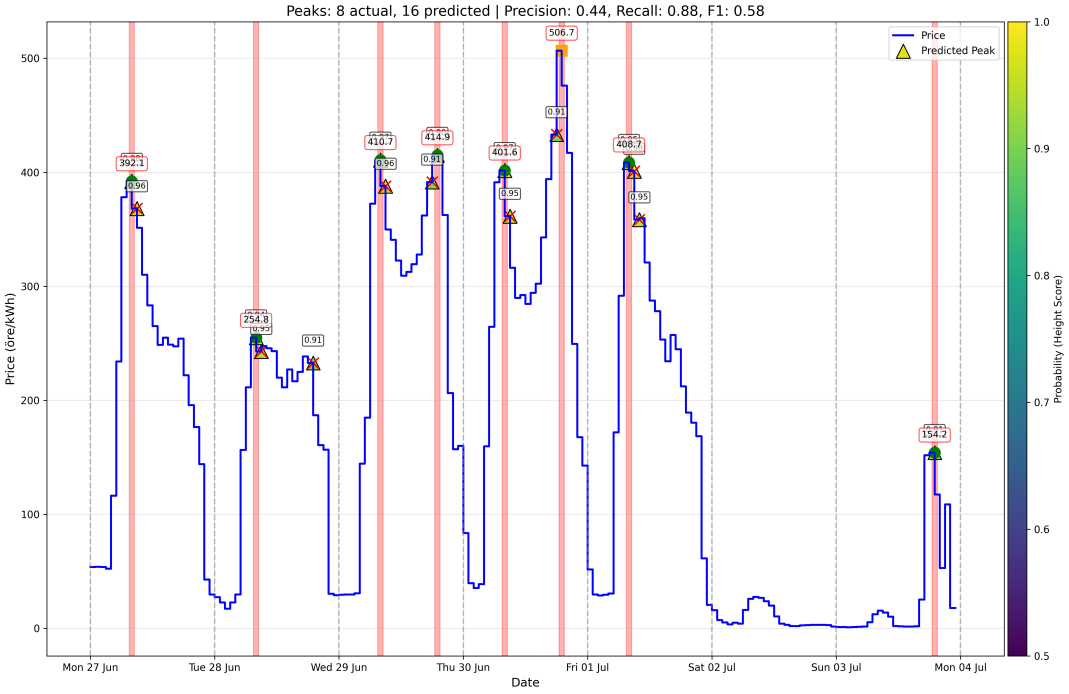
The resulting $F_1$-score is 0.58, reflecting the trade-off: the classifier is tuned to prioritize recall (so as not to miss rare high-price hours), at the expense of many false positives.

**Valleys**
Similarly, in Figure 4.3b we see the valley-detection performance over one week of labeled data. The blue line again denotes the hourly spot price, red triangles mark the actual (ground-truth) valley hours, and inverted green triangles mark the predicted valleys. The shaded vertical bands in this plot indicate actual valley periods.

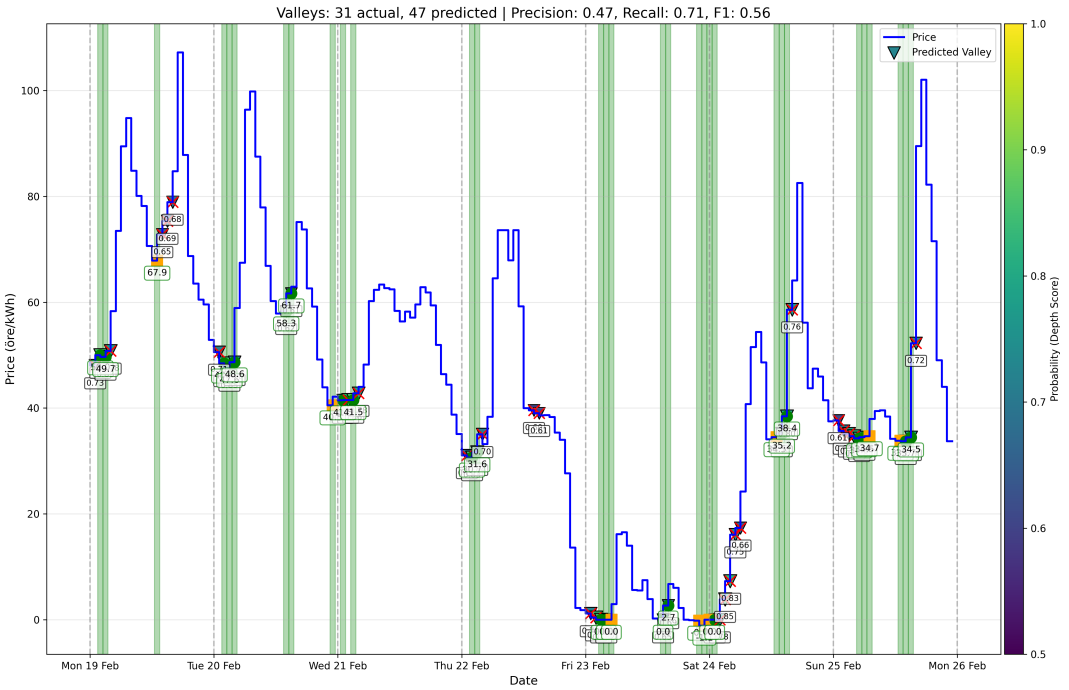
During this week, there were 31 true valley hours, but the model predicted 47. As a result, the precision is 0.47 (fewer than half of predicted valleys align with the ground truth) while the recall is 0.71 (about 70% of true valleys are successfully identified). The combined $F_1$-score of 0.56 reflects this balance, again emphasizing that the classifier is tuned to favor recall over precision.

Notice how several actual valley periods span multiple consecutive hours (for example, the trough around February 23–24), yet the model sometimes captures only a subset of those hours, or extends the predicted band into adjacent non-valley hours. This behavior underlines the difficulty of perfectly labeling extended low-price runs, while SMOTE-augmented training and threshold tuning help catch most valley events, occasional false positives and false negatives remain inevitable.

Visual inspection of these shaded bands thus remain crucial for validating whether the classifier's "imperfect but recall-focused" performance is acceptable before passing its signals to the RL controller.




**(a)** Peak Model performance over one week of labeled data.




**(b)** Valley Model performance over one week of labeled data.

**Figure 4.3:** Model performance for predicted peaks (top) and valleys (bottom) over one week of labeled data. (underestimates due to some true peaks missing in labels)

### 4.1.2 Price trend model

Figure 4.4 illustrates the Trend Model's performance on SE3 price data throughout April 2024. In the top panel, the blue curve shows the actual hourly spot prices, while the red curve depicts the model's 24-hour-ahead forecasts. We observe that the forecast generally captures broad daily patterns, such as gradual rises during mid-April volatility and lower prices in early April but smoothes out short-lived spikes and dips.

The middle panel plots the point-wise error, defined as the *actual* − *predicted*. Green bars above zero indicate hours when the model underestimates the actual price, and red bars below zero indicate overestimates. During periods of rapid price escalation we expect to see large errors since the trend model should capture the overal shape rather than the high volatility.

In the bottom panel, the orange bars represent daily MAE while the dashed blue line shows the average actual daily price, which closely mirrors the red forecast curve in the top panel, indirectly showing the model's overall accuracy.

Directional accuracy is calculated as 0.57, but this is misleading since the metric is calculated on too granular data. The average daily price metric provides a more meaningful performance measure, although here it is presented only in the visually.

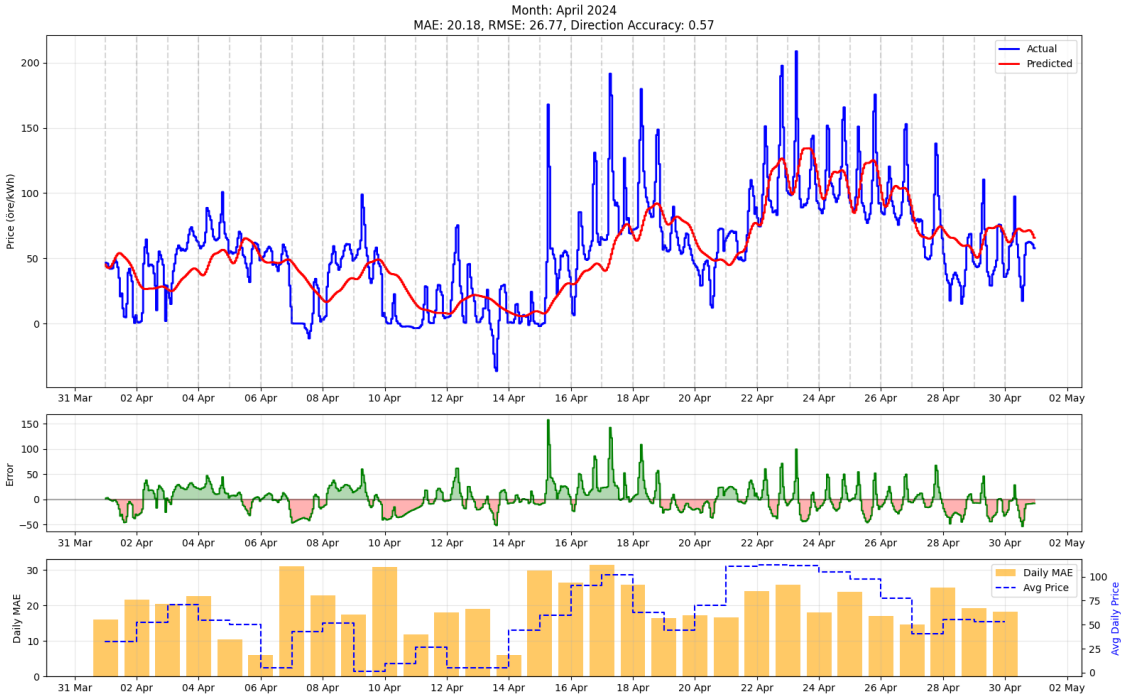



**Figure 4.4:** Trend Model performance for April 2024. Top: actual vs. predicted hourly SE3 prices (blue/red). Middle: hourly error (green = overestimate, red = underestimate). Bottom: daily MAE (orange) and average daily price (dashed blue). Metrics: MAE 20.18 öre, RMSE 26.77 öre, direction accuracy 0.57

### 4.1.3 Merged price model

The merged price model injects detected peaks and valleys into the Trend model's smooth forecast. Figure 4.9 shows this merging. Note that we use the classifier's probability score to set the spike height even though the peak/valley model isn't trained to predict actual price magnitudes. As a result, these injected peaks may not reflect true peak heights. By adding flags nonetheless, large errors around sharp spikes and drops are reduced compared to the Trend model alone.

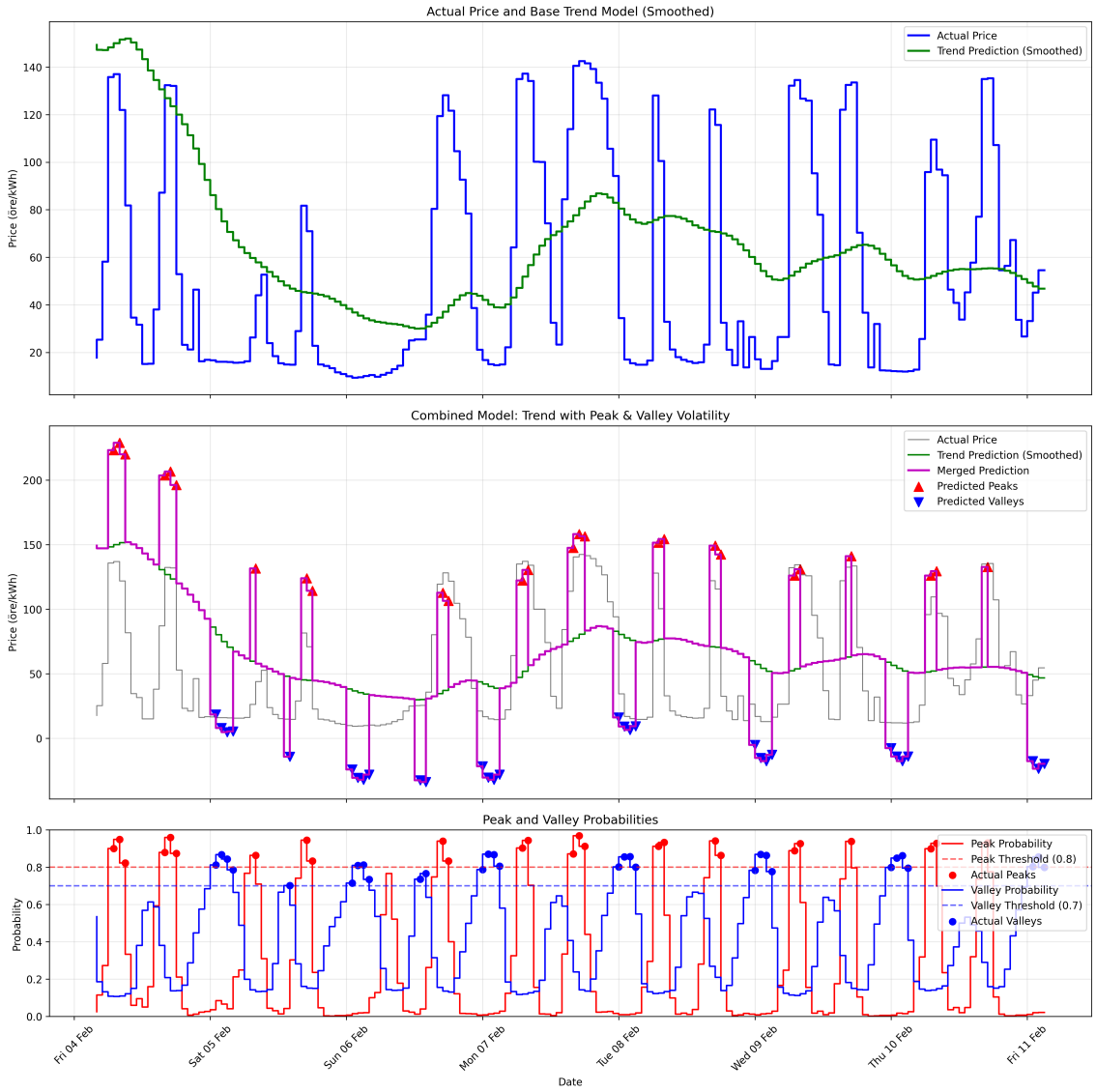



**Figure 4.5:** Merged model over one-week validation. Top: actual (blue) vs. trend forecast (green). Middle: merged output (magenta) with peaks (red) and valleys (blue). Bottom: classifier probabilities (thresholds shown).

## 4.2 Solar forecasts

Figure 4.6 compares hourly predicted (orange) and actual (blue) solar production for five consecutive days (March 15–19, 2025). Across this interval, the forecast closely tracks the classic ramp-up and ramp-down PV output. The hourly MAE over these five days is 0.45 kW (out of the 20.3 kW system), corresponding to an sMAPE of 5.3%. Minor overpredictions occur, but overall the timing and magnitude of peaks match closely the actual production.

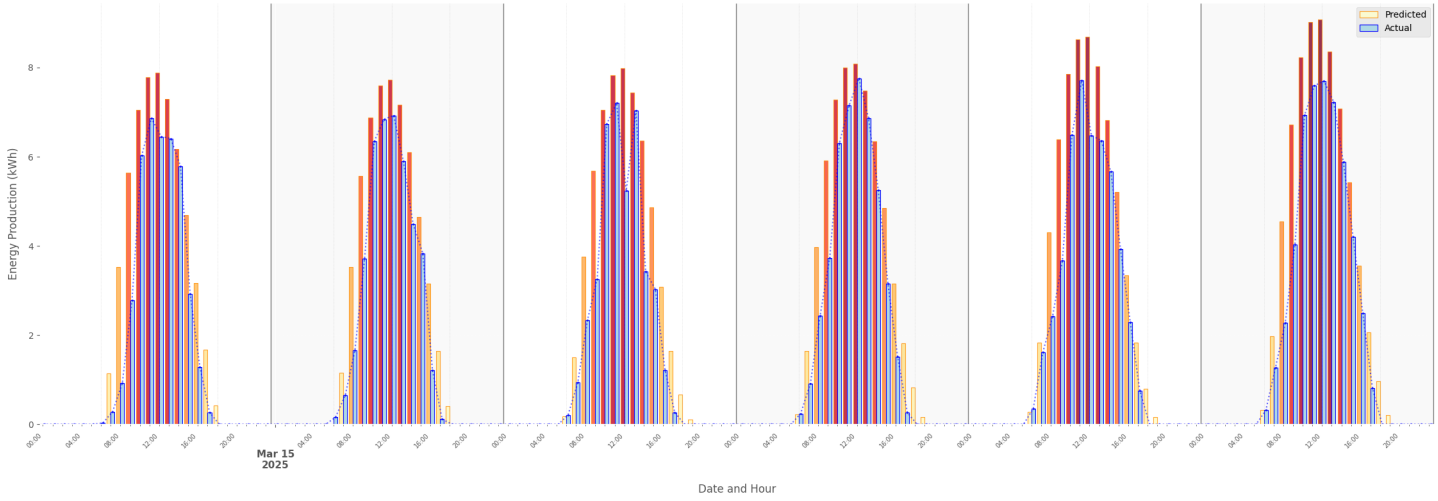



**Figure 4.6:** Hourly predicted (orange) versus actual (blue) solar energy production for the PV installation over five consecutive days (March 15–19, 2025), illustrating the close alignment of the forecasted and measured outputs.

Figure 4.7 presents a heatmap of predicted hourly production from March 1 to March 10, 2025. Each cell's color intensity indicates the forecasted kWh for that hour and date. Notice how the predictions captures the variability of cloud cover on March 4–5 (midday dips) and correctly shifts the noon-hour peak later on March 7 when sunrise was delayed. Over these ten days, the average daily predicted energy is 25.6 kWh.

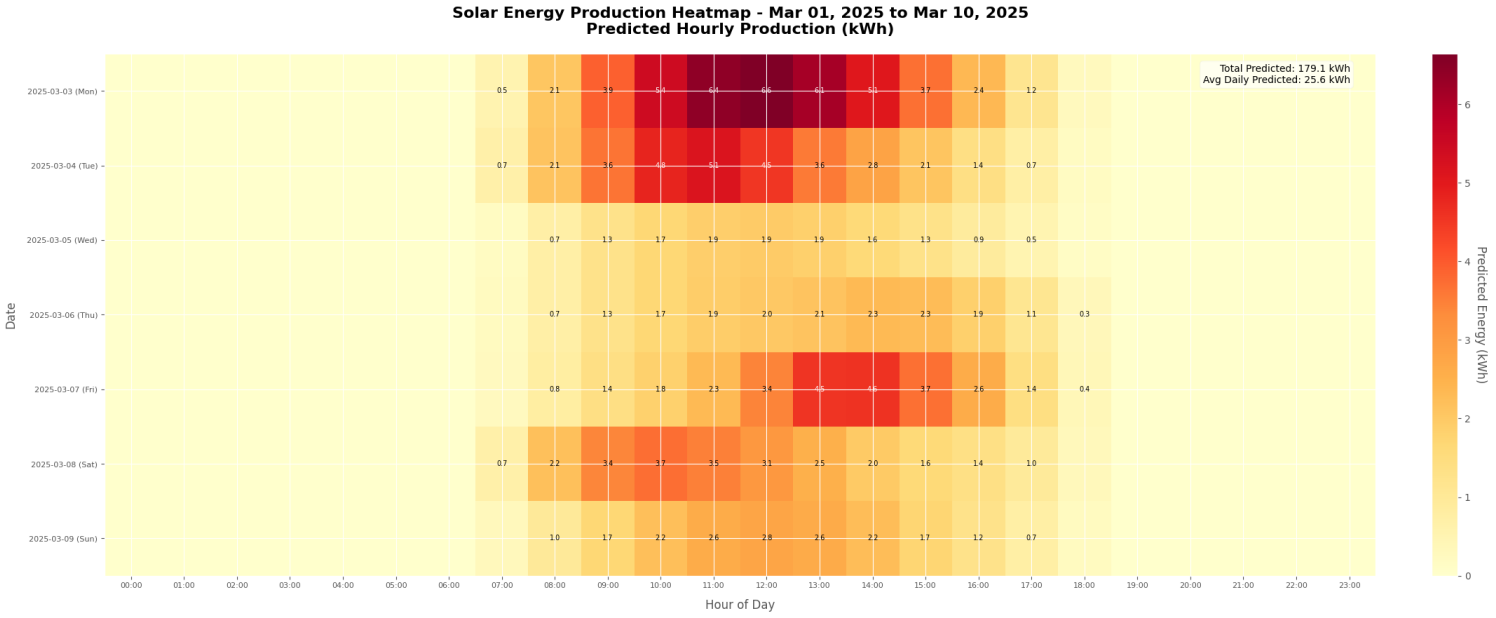



**Figure 4.7:** Heatmap of predicted hourly solar energy production from March 1 to March 10, 2025. Each cell shows the forecasted kWh for that hour and date, with deeper reds indicating higher output.

## 4.3 Demand Model

Figure 4.8 shows the Home Demand Model's performance from March 1–15, 2025.

In the top panel, actual hourly consumption (blue) and predicted demand (orange) are overlaid, with a shaded ±1 uncertainty band. Over these 336 hours, the model achieves RMSE = 0.80 kWh, MAE = 0.46 kWh, and $R^2$ = 0.874. Peaks from morning and evening heating loads are well captured.

The middle-left heatmap displays HMM-derived occupancy states (0 = low, 1 = medium, 2 = high) by hour and day, indicating higher load when occupancy is high.

The bottom-left timeline traces the continuous HMM state sequence, revealing weekday transitions. The bottom-right plot shows prediction residuals (actual – predicted), which mostly lie within ±1 kWh; larger errors occur mainly due to a lead or lag in the prediction.

Overall, the model tracks baseline patterns and heating-related peaks very accurately, with HMM states, temperature, and lag features enabling robust adaptation to both daily cycles and anomalies.

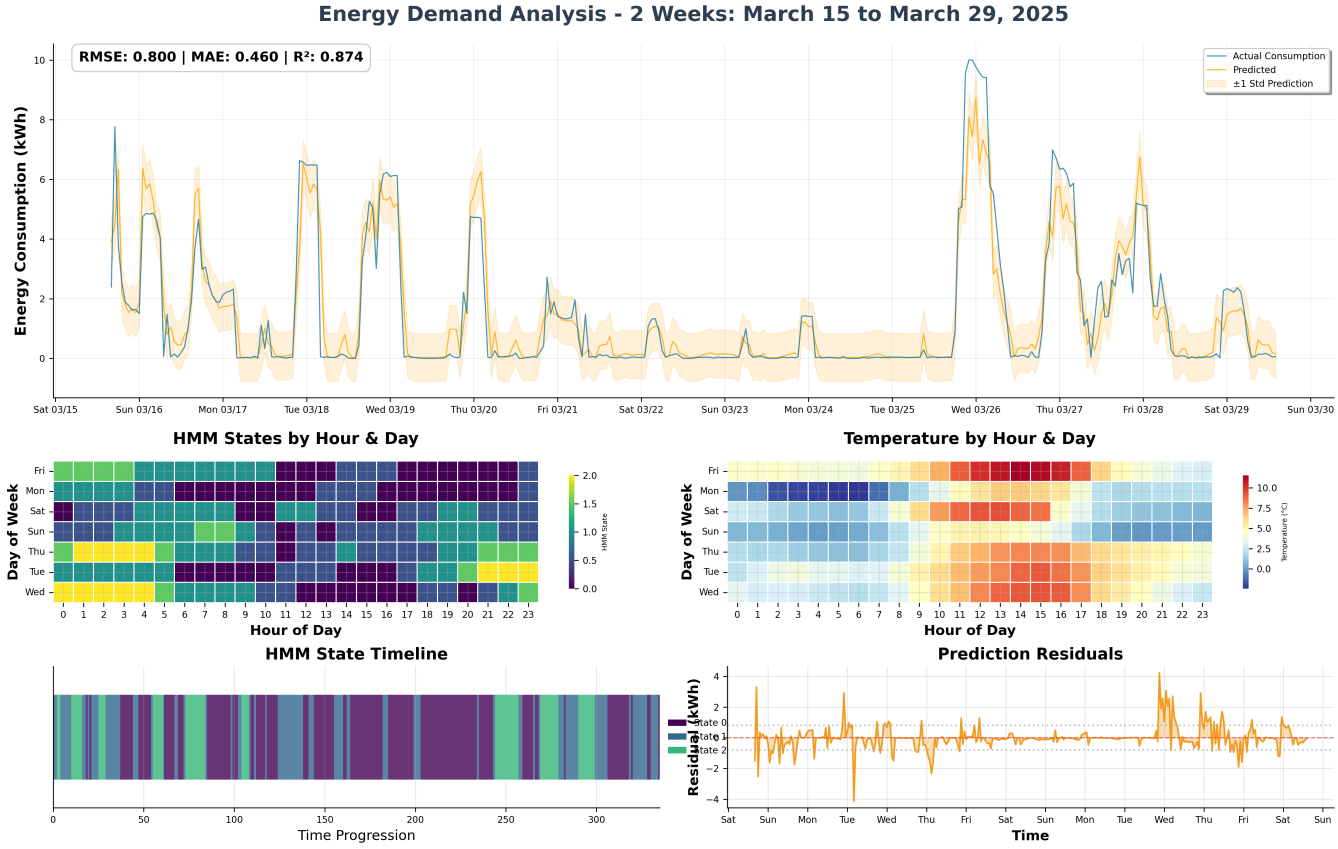



**Figure 4.8:** Home Demand Model performance. Top: actual vs. predicted consumption (±1$\sigma$). Middle: HMM occupancy states and ambient temperature by hour/day. Bottom: HMM state timeline and residuals.

Figure 4.9 ranks the Demand Model's top 20 features by their normalized XGBoost gain. The single most influential feature is "Hp Contribution" (heat-pump load), accounting for ≈ 19% of total gain.

Immediate consumption metrics "Consumption" (≈ 11%) and "Consumption Power Ratio" (≈ 10%) are next, highlighting the importance of current usage levels.

Weekday vs. weekend demand ("Consumption Dow (day-of-week) Avg Ratio," ≈ 9%) and HMM-inferred occupancy states ("HMM State Posterior 0," ≈ 1.7%; "HMM State Posterior 2," ≈ 1.5%) also rank highly, showing that both occupancy and day-of-week patterns matter.

Lagged consumption (e.g., "Consumption Pct 1H," "Consumption Same Hour 1D Ago," "Consumption Lag 24H") and solar–temperature interaction features (≈ 1% each) further refine the model, while lower gain features like multi-day rolling means (≈ 0.6%–0.8%) demonstrate diminishing returns.

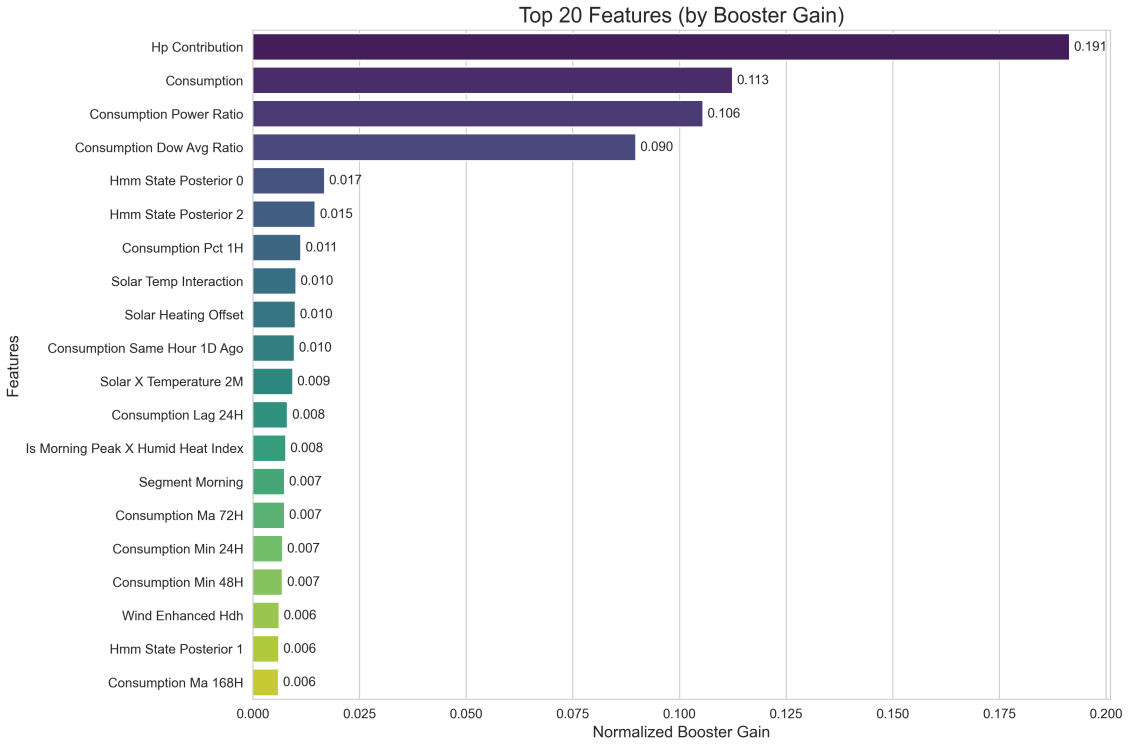



**Figure 4.9:** Top 20 features by XGBoost normalized gain in the Home Demand Model, highlighting heat-pump contribution, raw consumption metrics, occupancy and lag-based predictors among others.

## 4.4 Reinforcement Learning Agent

The RL-based controller demonstrates robust, data-driven battery management with clear improvements over conventional, rule-based baselines in our simulated environment. Training is performed with the RecurrentPPO algorithm; the final hyperparameters are summarized in Table 3.2. A full training run (30 million timesteps) on an AMD Ryzen 5700X CPU (no GPU) requires approximately 78 hours.

### 4.4.1 Performance Results

**Cost under low solar generation:** In the "cloudy month" scenario (January 2025), shown in Figure 4.10, the RL controller achieves the lowest total cost at 2 550 SEK, a 41 % reduction compared to no battery (4 320 SEK) and a 15 % improvement over the best rule-based method (Peak Shaving, 2 990 SEK) for this period. Although net profit is impossible when solar yield is minimal, the agent effectively limit grid imports to minimize monthly expense.

**Net benefit with generous PV:** In contrast, during a "sunny month" (May 2025, Figure 4.11), the same agent realizes a net benefit of 900 SEK, turning otherwise idle battery capacity into revenue. This marks a 125 % increase versus no battery (400 SEK) and a 50 % gain over Peak Shaving (600 SEK), this show the controller's ability to collect surplus solar and arbitrage price fluctuations.

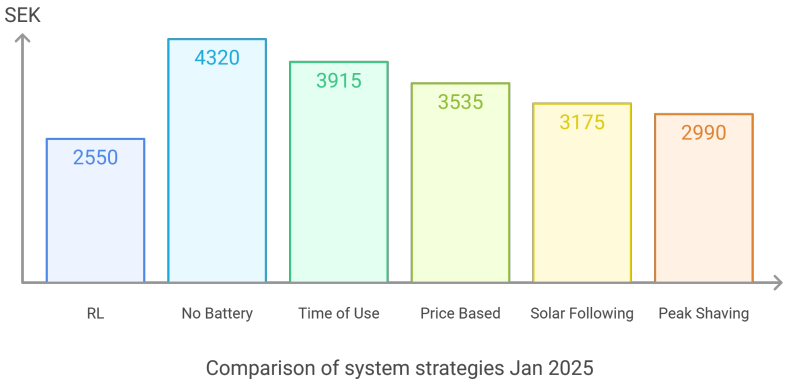



**Figure 4.10:** Monthly cost for different control strategies under a cloudy month (Jan 2025).

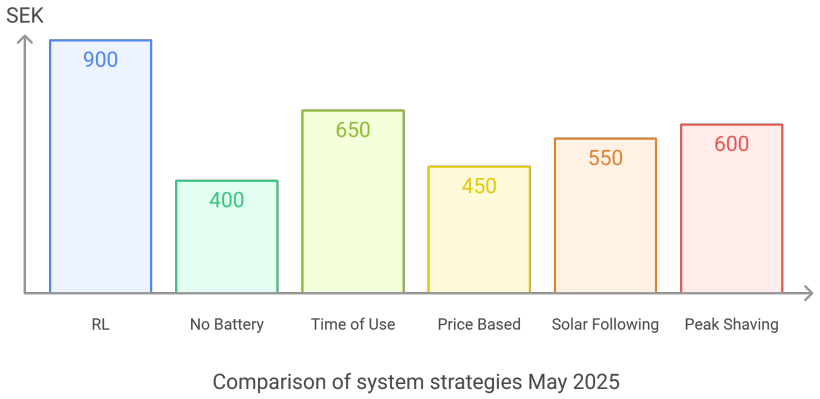



**Figure 4.11:** Net monthly benefit (SEK) for different control strategies under a sunny month (May 2025).

**Peak Reduction Performance:** Figure 4.12 clearly illustrates that the RL controller outperforms all baseline strategies in shaving peak grid import. With the RL policy, the highest hourly draw is limited to about 6.5 kW, compared to 11 kW when no battery is used (a 41 % reduction). The rule-based schemes all perform between roughly 9 and 10 kW, Time-of-Use at $\tilde{9}.98$ kW, Price-Based and Solar-Following both at $\tilde{9}.01$ kW, and Peak-Shaving interestingly at 7 kW, showing that even dedicated peak-shaving logic cannot match the learned, price and forecast-aware behavior of the RL agent (although the peak strategy only being a fixed "dump battery when import over 6 kW", not ensuring available battery power for those occasions.). This reduction in peak demand directly translates into lower capacity charges and greater operational flexibility, demonstrating the RL approach's ability to anticipate and pre-empt high-price/high-demand hours.

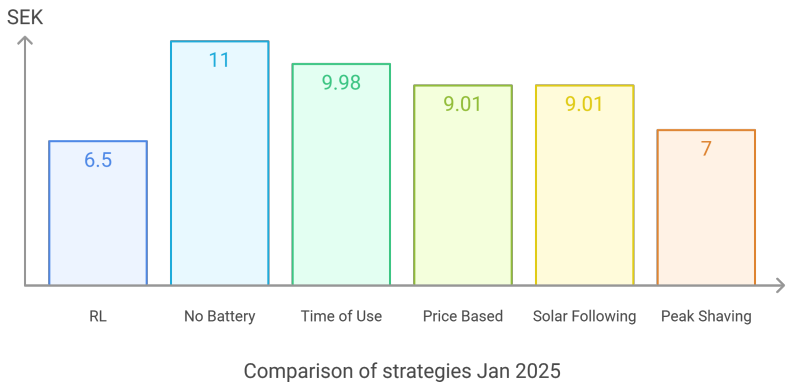



**Figure 4.12:** Maximum hourly import (kW) for different control strategies in January 2025.

### 4.4.2 Battery Operation and SoC Dynamics

Figure 4.13 illustrates a 9-day interval of the agent's operation (June 2024). The top panel overlays battery state-of-charge (SoC) against the hourly electricity price. The agent charges aggressively during low-cost windows and discharges when prices exceed a learned threshold. The middle panel shows household load and PV output. From these traces we observe that the agent maintains SoC within preferred bands (15–85 %) while exploiting both price arbitrage and solar surplus.

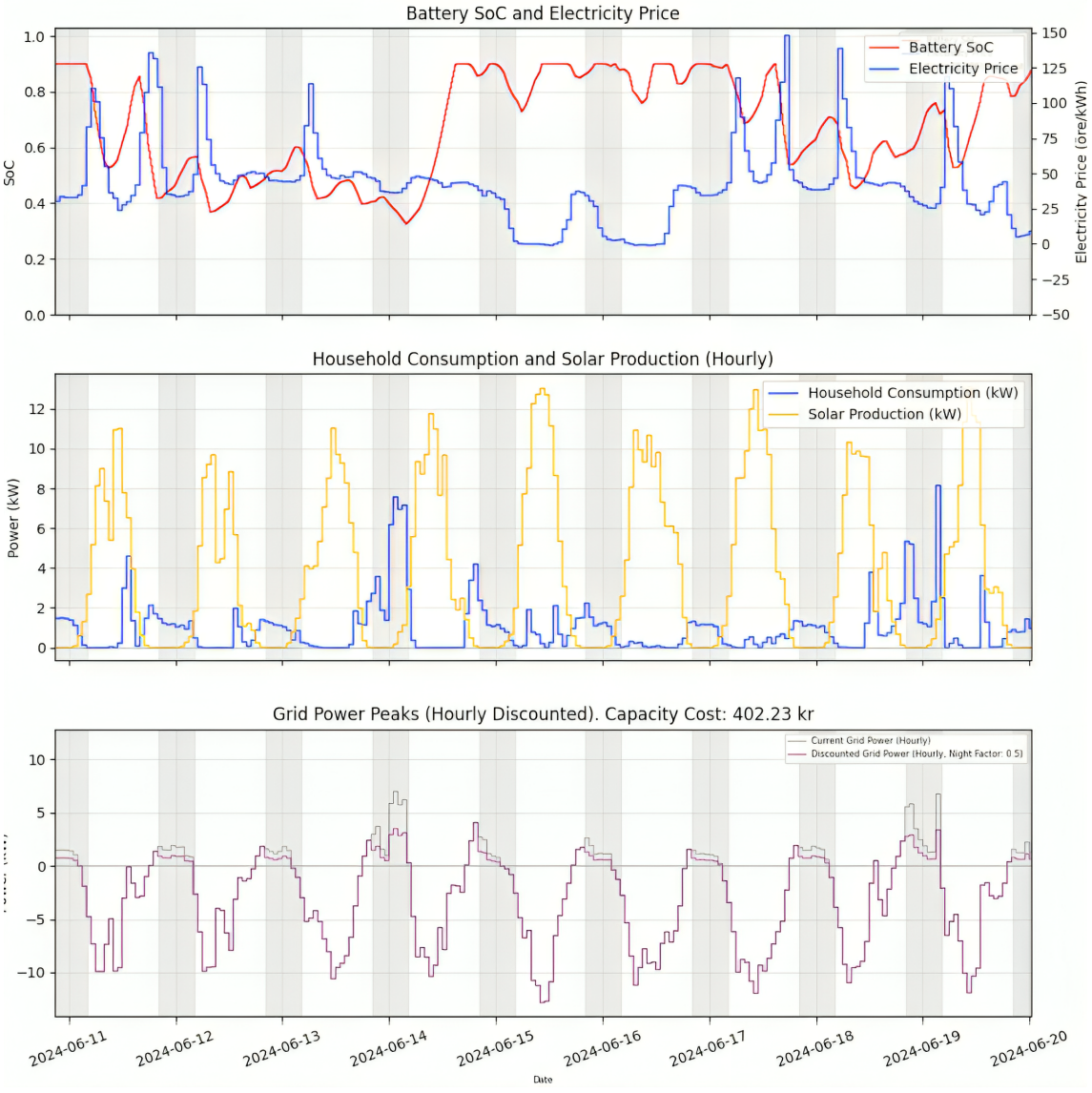



**Figure 4.13:** Ten-day battery operation: SoC vs. electricity price (top), household consumption and PV production (middle), and hourly grid import with discount factor (bottom). Gray shading denotes nighttime discount periods.

# 5
# Conclusion

This thesis demonstrates the feasibility and potential of AI-driven home energy management systems in the context of Sweden's transition to power-based electricity tariffs. Despite the complexity of integrating forecasting models, reinforcement learning control, and real-time hardware operation, the results show promising progress toward intelligent, automated battery management.

The developed system successfully coordinates multiple dynamic factors, spot price fluctuations, solar production variability, household consumption patterns, and power-based fees into a unified control strategy. Field results confirm that AI approaches can yield measurable economic benefits, even with limited training time and minimal hyperparameter tuning. These findings suggest that further model refinement, extended training periods, and improved reward design could significantly enhance performance.

Today's residential battery management still shows a large gap between available technology and practical use. Most homeowners either rely on built-in manufacturer modes that ignore price signals, or manually schedule charging and discharging based on forecasts. The former neglects profit and grid optimization entirely, while the latter demands expertise and time for suboptimal results. Neither approach accounts for battery wear, dynamic grid tariffs, or coordinated solar-battery optimization.

While rule-based control remains suitable for simple applications, it lacks adaptability to the multi-objective optimization challenges of modern energy systems. In contrast, the reinforcement learning model developed here demonstrates the ability to recognize temporal patterns, adapt to changing conditions, and autonomously execute strategies such as pre-emptive discharging before solar peaks. Once deployed, it operates independently, removing the need for user intervention.

The system's ability to maintain safe operation while improving grid independence shows that AI-driven control can be both practical and reliable. Even with limited training, it achieved meaningful cost reduction and peak-shaving performance. With additional optimization, it could further reduce household costs and alleviate grid strain through intelligent scheduling and energy arbitrage.

## 5.1 Ethical and Environmental Considerations

Home Energy Management Systems like the one developed in this project rely on capital-intensive technologies such as solar panels and batteries. As a result, they risk amplifying socioeconomic disparities by primarily benefiting homeowners with financial means, while renters and low-income users remain more exposed to rising electricity tariffs. To counteract this transparent reporting should be prioritized to ensure inclusivity and avoid reinforcing structural bias.

Training AI models can be energy-intensive, raising valid environmental concerns. However, in this project, the energy required for model training was marginal compared to the expected long-term savings achieved by the control system. The overall environmental impact is therefore positive, though such assessments must always be made case by case.

Battery and photovoltaic production also introduce sustainability challenges due to material extraction, emissions, and end-of-life waste. A complete life-cycle assessment is essential to evaluate their true environmental cost, emphasizing recycling and responsible material recovery.

Looking forward, this project provides a strong base for developing more advanced residential energy systems. Future improvements include longer model training, better fine-tuning for more convergence, and integration of additional controllable loads like electric vehicles and washing machines.

As Sweden transitions to universal power-based tariffs by 2027, the demand for intelligent energy management will continue to grow. This thesis shows that AI-driven control can meet that challenge, delivering both individual economic savings and broader benefits for the electrical grid. While further development is required for large-scale adoption, the technical foundation is clear. Machine learning can transform home batteries from passive backup units into active participants in the sustainable energy transition.

# Bibliography


[1] Ellevio AB, "Ny prismodell baserad på effekt," *Ellevio*, [Online]. Available: https://www.ellevio.se/abonnemang/ny-prismodell-baserad-pa-effekt/ [Accessed: May 25, 2025].

[2] Energimarknadsinspektionen, "Effektavgift," *Energimarknadsinspektionen*, [Online]. Available: https://ei.se/konsument/el/elnatsavgiften-och-elnatsreglering/effektavgift [Accessed: May 25, 2025].

[3] S. Stavrev and D. Ginchev, "Reinforcement Learning Techniques in Optimizing Energy Systems," *Electronics*, vol. 13, no. 8, p. 1459, Apr. 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/8/1459 [Accessed: May 25, 2025].

[4] L. Tschora, "Machine Learning Techniques for Electricity Price Forecasting," Ph.D. dissertation, INSA de Lyon, Lyon, France, 2024. [Online]. Available: https://theses.hal.science/tel-04728900/file/these.pdf [Accessed: May 25, 2025].

[5] Agency for the Cooperation of Energy Regulators (ACER), "Report on Electricity Transmission and Distribution Tariff Methodologies in Europe," ACER, Luxembourg, 2023. [Online]. Available: https://www.acer.europa.eu/sites/default/files/documents/Publications/ACER_electricity_network_tariff_report.pdf [Accessed: May 25, 2025].

[6] M. A. Khan, M. A. Khan, and M. A. Khan, "Applications of Deep Reinforcement Learning for Home Energy Management Systems: A Review," *Energies*, vol. 17, no. 24, p. 6420, Dec. 2024. [Online]. Available: https://www.mdpi.com/1996-1073/17/24/6420 [Accessed: May 25, 2025].

[7] S. Ibrahim, M. Mostafa, A. Jnadi, H. Salloum, and P. Osinenko, "Comprehensive Overview of Reward Engineering and Shaping in Advancing Reinforcement Learning Applications," arXiv:2408.10215 [Preprint], 2024. [Online]. Available: https://arxiv.org/abs/2408.10215 [Accessed: May 25, 2025].

[8] Energimarknadsbyrån, "Energiskatt – skattesatser och kostnader," *Energimarknadsbyrån*, [Online]. Available: https://www.energimarknadsbyran.se/el/dina-avtal-och-kostnader/elrakningen/energiskatt-skattesatser-och-kostnader/ [Accessed: May 25, 2025].

[9] Energimarknadsinspektionen, "Energimarknadsinspektionens föreskrifter och allmänna råd för utformning av nättariffer för ett effektivt utnyttjande av elnätet, EIFS 2022:1," Energimarknadsinspektionen, 26 Apr. 2022. [Online]. Available: https://ei.se/download/18.b0dbdc118002bc176c133ae/1650953845317/EIFS-2022-1-om-utformining-av-n%C3%A4ttariffer-f%C3%B6r-ett-effektivt-utnyttjande-av-eln%C3%A4tet.pdf [Accessed: May 25, 2025].

[10] Sonnen Australia, "Technical Documents," *Sonnen Australia*, [Online]. Available: https://sonnen.com.au/technical-documents/ [Accessed: May 25, 2025].

[11] Alternative Energy Tutorials, "Solar Irradiance," Alternative Energy Tutorials, [Online]. Available: https://www.alternative-energy-tutorials.com/solar-power/solar-irradiance.html [Accessed: May 25, 2025].

[12] Penn State University, "Utility Solar Power and Concentration," Penn State e-Education, [Online]. Available: https://www.e-education.psu.edu/eme812/node/896 [Accessed: May 25, 2025].

[13] M. Ben Amara et al., "Performance modeling and investigation of fixed, single and dual-axis tracking photovoltaic panel in Monastir city, Tunisia," *Renewable and Sustainable Energy Reviews*, vol. 15, pp. XYZ–XYZ, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1364032111002772 [Accessed: May 25, 2025].

[14] GeeksforGeeks, "Activation Functions in Neural Networks," GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/activation-functions-neural-networks [Accessed: May 25, 2025].

[15] J. Yang, Y. Wang, and X. Li, "Prediction of stock price direction using the LASSO-LSTM model combining technical indicators and financial sentiment analysis," *PeerJ Comput. Sci.*, vol. 8, p. e1148, 2022, doi: 10.7717/peerj-cs.1148.

[16] Prefect, "Prefect Documentation," *Prefect*, [Online]. Available: https://docs.prefect.io/v3/get-started [Accessed: May 25, 2025].

[17] Nord Pool Group, "EUPHEMIA Public Description," Nord Pool Group, 2024. [Online]. Available: https://www.nordpoolgroup.com/globalassets/download-center/single-day-ahead-coupling/

euphemia-public-description.pdf [Accessed: May 25, 2025].

[18] J. Lago, F. De Ridder, P. Vrancx, and B. De Schutter, "Forecasting day-ahead electricity prices in Europe: The importance of considering market integration," arXiv:1708.07061 [Preprint], 2017. [Online]. Available: https://arxiv.org/abs/1708.07061 [Accessed: May 25, 2025].

[19] Forecast.solar, "Forecast.solar," Forecast.solar, [Online]. Available: https://forecast.solar/ [Accessed: May 25, 2025].

[20] Farama Foundation, "Gymnasium," Farama Foundation, [Online]. Available: https://gymnasium.farama.org/ [Accessed: May 25, 2025].

**DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY**
**CHALMERS UNIVERSITY OF TECHNOLOGY**
Gothenburg, Sweden
www.chalmers.se


CHALMERS
UNIVERSITY OF TECHNOLOGY