Closed-Loop Reinforcement Learning for Short-Term Load Forecasting over a REST API Framework

Julien Guité-Vinet, Alexandre Blondin Massé, Éric Beaudry, Arnaud Zinfou

Keywords: short-term load forecasting, reinforcement learning, closed-loop design, model lifecycle, REST API

Summary

Time series forecasting is a crucial task in various domains because it allows for the forecast of future values based on historical data. This is essential for making informed decisions in fields such as finance, energy, retail, meteorology, and many others. Accurate forecasting can help optimize operations, reduce costs, and improve planning and resource allocation. During the lifecycle of a model, it is important to manage the lifecycle of models, including training, validation, and hyperparameter tuning. In this paper, we propose a novel approach to short-term load forecasting using reinforcement learning in a closed-loop manner to optimize the lifecycle of models. We evaluate the proposed approach using a real-world energy dataset and a REST API framework. The results demonstrate an improvement of 0.77% in the baseline MAPE score while minimizing the need for manual intervention in managing the lifecycle of forecasting models. This highlights the system's ability to autonomously adapt to changing conditions and optimize model performance over time.

Contribution(s)

1. We introduce a closed-loop system that enables a reinforcement learning agent to effectively manage the lifecycle of forecasting models.
   Context: While prior researches have explored closed-loop systems, the lifecycle management of forecasting models has not been explicitly addressed.
2. We propose a novel reward schema tailored to the lifecycle management of forecasting models.
   Context: Previous studies have utilized reward schemas to train reinforcement learning agents for optimizing short-term load forecasting models. However, these approaches have not incorporated the lifecycle management of forecasting models.
3. We extend the REST API framework to function as a decision-making system for short-term load forecasting.
   Context: Existing work on reinforcement learning in closed-loop systems has not explicitly considered the REST API framework as an environment for decision-making.

Closed-Loop Reinforcement Learning for Short-Term Load Forecasting over a REST API Framework

Julien Guité-Vinet[1], **Alexandre Blondin Massé**[1,2], **Éric Beaudry**[1], **Arnaud Zinfou**[2]
{guite-vinet.julien,blondin.masse.alexandre,beaudry.eric}@uqam.ca,
zinfou.arnaud@hydroquebec.com

[1]**Department of Computing Science, Université du Québec à Montréal, Canada**
[2]**Institut de Recherche d'Hydro-Québec, Canada**

Abstract

Time series forecasting is a crucial task in various domains because it allows for the forecast of future values based on historical data. This is essential for making informed decisions in fields such as finance, energy, retail, meteorology, and many others. Accurate forecasting can help optimize operations, reduce costs, and improve planning and resource allocation. During the lifecycle of a model, it is important to manage the model's lifecycle, including training, validation, and hyperparameter tuning. In this paper, we propose a novel approach to short-term load forecasting using reinforcement learning in a closed-loop manner to optimize the lifecycle of models. We evaluate the proposed approach using a real-world energy dataset and a REST API framework. The results demonstrate an improvement of 0.77% in the baseline MAPE score while minimizing the need for manual intervention in managing the lifecycle of forecasting models. This highlights the system's ability to autonomously adapt to changing conditions and optimize model performance over time.

1 Introduction

Time series forecasting is a canonical problem with applications covering wide-ranging domains including retail, meteorology, economics, epidemiology and energy. Accurate forecasting is essential to optimize operations and reduce costs. Many different approaches have been proposed for short-term load forecasting, including autoregressive models, moving average models, exponential smoothing models, and deep learning models (Hyndman, 2018).

Finding the optimal model for a given situation is a challenging task. The optimal model may change over time as the underlying data distribution changes. In practice, it is difficult to know which model will perform best for a given situation. This is especially true in the context of short-term load forecasting, where the underlying data distribution can be complex and non-stationary. For instance, energy demand may show seasonal variations, with higher demand during the winter months compared to the summer. As a result, this could necessitate distinct models with varying performance across different seasons, or alternatively, a single model that performs well in the summer but demonstrates suboptimal performance in the winter. Thus models follow a lifecycle that involves several stages, each of which presents unique challenges.

These stages are typically categorized as problem definition, data collection, preprocessing, model design and training, evaluation and testing, deployment, and maintenance. The maintenance stage involves monitoring the model's performance over time, adapting to changes in data distribution, and refitting the model when necessary (Miao et al., 2017). Over time, the distribution of incoming data may shift, rendering the model less effective. This phenomenon, known as concept drift, necessitates regular monitoring and refitting.

The formalism of reinforcement learning (RL) is well-suited for this kind of sequential decision-making tasks where the effects of actions are not immediately observable. It can also be naturally framed into a representational state transfer (REST) application programmable interface (API) paradigm. A REST API is an application programming interface that conforms to the REST architecture, which is a stateless, client-server, cacheable, and uniform interface. It allows different systems to communicate with each other over the internet using standard protocols. A REST API framework consists of resources, which are the objects or data that can be accessed through the API. Resources are identified by uniform resource identifiers (URI) and can be manipulated using standard HTTP methods such as GET, POST, PUT, and DELETE (Richardson & Ruby, 2008).

In this paper, we propose a novel approach to short-term load forecasting using reinforcement learning in a closed-loop manner to optimize the lifecycle of models using a REST API. Our contributions are threefold:

- We propose a closed-loop system that allows a RL agent to manage the lifecycle of models,
- We design a reward schema for the lifecycle of models,
- We specialize the REST API into a decision-making system for short-term load forecasting.

This paper is organized as follows. In Section 2, we introduce the preliminary concepts. In Section 3, we formulate the problem. In Section 4, we summarize work related to model forecasting in a closed-loop manner. In Section 5, we describe our approach. In Section 6 we present the experimental results. Finally, in Section 7, we conclude and discuss future work.

2 Preliminaries

A time series is defined as a sequence of data points indexed in chronological order. Load time series can generally be decomposed into two primary components: trend and seasonality, which represent respectively the overall direction and cyclic patterns of the data (Deng & Jirutitijaroen, 2010). Short-term load forecasting involves predicting future values based on historical data over a short-term horizon, typically ranging from a few hours to one week (Ullah et al., 2024). Additionally, exogenous variables, such as temperature or holiday periods, are external factors that influence the load demand. For instance, lower temperatures may increase demand due to heating requirements, while holidays may reduce industrial activity, thereby reducing demand (Yasuoka et al., 2004). Models are commonly evaluated using metrics such as mean square error (MSE) or the mean absolute percentage error (MAPE).

Reinforcement learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with an environment. The agent takes actions in the environment and receives rewards based on the actions taken on his current state. The goal of the agent is to learn a policy that maximizes the expected cumulative rewards. It follows a Markov decision process (MDP) formalism defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition probability function, $\mathcal{R}$ is the reward function, and γ is the discount factor.

Proximal Policy Optimization (PPO) is a policy gradient algorithm that accommodates continuous action spaces by parameterizing a stochastic policy over continuous distributions (Schulman et al., 2017). PPO supports continuous state representations via function approximators, typically neural networks, which map continuous inputs to policy and value function parameters. PPO also constrains the policy updates by optimizing a clipped surrogate objective function, thereby mitigating the risk of large, destabilizing policy shifts during training. Formally, this clipped objective is expressed as:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] - \beta\mathcal{H}(\pi_\theta) \tag{1}$$

where

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the current and old policies,
- $\hat{A}_t$ is the advantage estimate at time step t,
- $\mathcal{H}(\pi_\theta)$ is the entropy of the policy, which encourages exploration by a factor β.

The clipping function ensures that the policy update is only accepted if the probability ratio stays within the range $[1-\epsilon, 1+\epsilon]$, preventing overly large updates. The choice of the clipping parameter ϵ controls the trade-off between exploration and stability.

3 Problem Formulation

The task is to demonstrate that the agent learns a model management policy that improves over time over a long period. The agent must learn to select the best model for the current situation, refit models when necessary or keep its current model in order to improve forecasting accuracy based on the MAPE score expressed as :

$$\text{MAPE} = \frac{100}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{2}$$

where y_i is the true value and $\hat{y}_i$ is the forecasted value.

These various decision-making processes follow the form of a MDP with rewards defined as follows:

- Environment $\mathcal{E}$: The API system including the sets of available models and time series.
- State space $\mathcal{S}$: The current state of the agent, *i.e* the weekday, the hour, the temperature and its derivative, the forecasting model's name, last performance and age.
- Action space $\mathcal{A}$: The actions the agent can take, *i.e* select a model, refit a model or keep is current model on a hourly time step.
- Reward function $\mathcal{R}$: The reward the agent receives for taking an action and using the underlying model to make a forecast for a given horizon.
- Transition probability function $\mathcal{P} : \mathcal{A} \times \mathcal{S} \to \mathcal{S}$: The probability of transitioning from one state to another based on the action taken.
- Discount factor γ: The discount factor determines the importance of future rewards.

The goal of the agent is to learn a policy that maximizes the expected cumulative reward over time, *i.e* to minimize the MAPE score by selecting the best model for the current situation.

4 Related Work

Many different approaches have been proposed to improve short-term load forecasting accuracy. Traditional methods include autoregressive models, moving average models, and exponential smoothing models (Soares & Medeiros, 2008; Cancelo et al., 2008; Clements et al., 2016). Deep learning (DL) models have been proposed, including recurrent neural networks and long short-term memory networks (Neil et al., 2016). More recently, DL models have been shown to outperform traditional methods in terms of accuracy (Oreshkin et al., 2019; Salinas et al., 2020; Lim et al., 2021; Zhou et al., 2023). In practice, DL models can be difficult to hypertune due to their high dimensionality, non-convex optimization landscapes, and sensitivity to initial conditions (Liao et al., 2022).

Hybrid closed-loop models are forecasting models that combine multiple types of models within a closed-loop architecture. A closed loop is a feedback system where the results or outputs of a process are fed back as inputs to improve or adjust the process. This allows the system to adapt and correct itself based on observed errors or performance (Mayne, 1999).

Zhang & Li (2020) align the task of clustering groups with an overall forecasting objective, thereby creating a closed loop. Specifically, regional load demand over a period can be expressed as the sum of the load of k groups of customers organized in a hierarchical structure. Finding an optimal k value can be challenging without domain knowledge as the groups are not determined based on the criterion of improving forecasting performance.

Other authors propose a similar method based on reinforcement learning, utilizing a deep Q-learning approach that follows a Markov decision process where an agent must assign a cluster to customers based on their load patterns (Zhang et al., 2021). A closed loop is formed by using the MAPE score of the forecasting models in the reward function.

Another closed-loop model has been proposed for multi-horizon short-term load forecasting, where forecasts are adapted based on results obtained over different horizons (Nejati et al., 2023). A feedback process measures the discrepancy between short-horizon and long-horizon forecasts. This measure then serves as a signal to form a closed-loop mechanism capable of dynamically adjusting short-term load forecasts.

Prediction intervals (PI) provide a means to quantify forecast uncertainty, but their quality depends on the ability to adapt to changes in data distributions. Zhang et al. (2022) propose a method based on deep Q-learning where the agent's objective is to minimize the width of the PI while maintaining the required coverage probability. A closed loop is formed by using the Winkler score of the forecasting model to reward the agent and balance the trade-off between PI width and coverage probability.

Work has been done to extend the REST API framework into a decision-making system, where the agent can interact with the API to make decisions based on the current state of the system. The research in this domain focuses on testing API (Kim et al., 2023; Nguyen et al., 2024; Corradini et al., 2024), optimizing API selection (Mandal et al., 2024) or optimizing the service interoperability with internet of things (IoT) (Kotstein & Decker, 2019).

OpenSTEF is an open-source software platform designed for transient stability analysis of power systems using a modular, microservice-based architecture (Garcia et al., 2025). While OpenSTEF's microservice-based architecture is well-suited for large-scale power system applications requiring modularity and scalability, many components must be customized manually or developed from scratch by the user, which limits its usability. In such cases, developing a tailored microservice may offer a more lightweight and efficient alternative.

5 Proposed Approach

We propose a RL layer aimed at managing the lifecycle of available forecasting models over a REST API framework environment, since the underlying structure of a REST API makes it well-suited for a closed-loop model, where the agent can interact with the API to manage the lifecycle of models.

In our experiment, a RL agent is responsible for maintaining the forecasting models in an offline system, where all data required is pre-collected and made available prior to execution. Initially, the agent must send a request to the API to fit a model, which returns a unique model identifier required to request a forecast to the API. For subsequent forecasts requests, the agent can either KEEP the same model identifier, FIT a different model, or SELECT another previously fitted model. The selection of fitted models is constrained to a maximum of n models, where we set $n = 10$ during our experimentation. When that limit is reached, the agent delete the oldest fitted model. Additionally, the agent is prohibited from fitting a model if it has been previously fitted within the past h hours, where $h = 6$ during our experimentation. At the end of each episode the agent reset its environment and the API is reset to its initial state. Table 1 presents the API routes used by the agent to manage the lifecycle of the models.

The `GET /scenario/{id}` route returns the scenario forecasts and can also return the MAPE score if the target time series true values are available. Since we use an offline system, we set the parameters of the `POST /models/fit/{m}?params` route to fit a model within a date range so

Table 1: Main endpoints used by the agent within its environment

Route	Description
`DELETE /models/fitted`	Delete all fitted models
`DELETE /models/fitted/{id}`	Delete fitted model `id`
`DELETE /scenarios`	Delete all forecasting scenarios
`DELETE /timeseries`	Delete all time series
`GET /models/fitted`	Get all fitted models
`GET /scenarios/{id}`	Get forecasting results of scenario `id`
`POST /models/fit/{m}?params`	Fit a parameterized model `m`
`POST /scenarios?params`	Create a parameterized forecasting scenario
`PUT /models/{m}`	Upload model `m`
`PUT /timeseries/{ts}`	Upload timeseries `ts`

that it is prohibited to see the target and exogenous time series values past one day prior the current datetime.

The reward function is defined as the negative of the MAPE score. However, the MAPE score is not enough to reflect the cost of taking an action. We extend the reward schema as follows:

- If the agent keeps the current model, the reward is 0,
- If the agent selects another model, the reward is -0.01,
- If the agent fits a model, the reward follows Equation 3 described below.

The fitting reward function should approach infinity as the MAPE score decreases toward zero, and exhibit an asymptotic decay toward zero as the MAPE score increases without bound. Furthermore, the reward function should incorporate a penalty mechanism that discourages the agent from fitting a model whose new performance is worst than before. The fitting reward function is defined as follows:

$$R_{fit}(x, b) = -\left(10 \log_{10}\left(\frac{x}{\phi}\right)\right) - \eta \left(\frac{x}{b}\right)^{\lambda} \tag{3}$$

where x is the current model MAPE score, ϕ is a reference score, b is the previous MAPE score of the same model, η is a penalty factor, and λ is a parameter that controls the decay of the reward function. The MAPE score is well-suited for this reward function as it enables a reference score of 100. In our experiment we set $\eta = 25$ and $\lambda = 4$. Figure 1 presents how the function behaves under different b values.

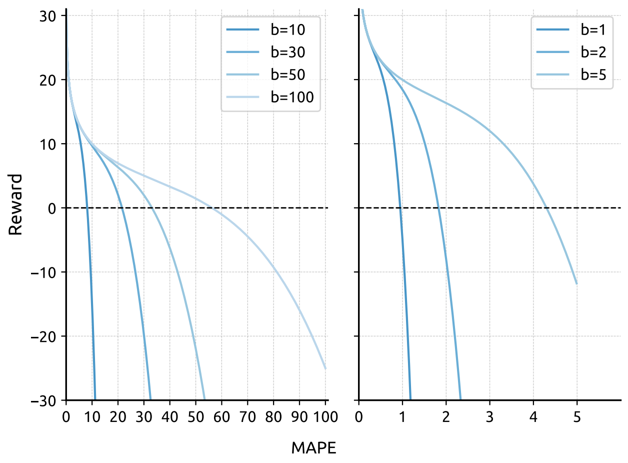


Figure 1: Reward function for the FIT action under different b values.

To avoid introducing bias during forecasting, we set up an experiment to evaluate the quality of learning over time by simulating the arrival of data in chronological order. The MAPE score of the models are used to update the agent, thus creating a closed-loop. We use the PPO algorithm to train the agent, which learns a policy that maximizes the expected cumulative reward. Figure 2 presents the process of the system.

$$\begin{tikzcd}
\mathcal{Y}(t) \arrow[r] & \text{models} \arrow[loop above, "\text{learns}"] \arrow[r] & \text{evaluation} \arrow[r] & \text{Agent} \arrow[loop above, "\text{learns}"] \arrow[ll, bend left, "\text{action}"]
\end{tikzcd}$$

Figure 2: Closed-loop reinforcement learning methodology for short-term load forecasting where $\mathcal{Y}(t)$ is the time series. The agent adapts his policy using the performance of the models as a reward.

6 Experimental Scenarios

In this section, we present the collected results of our agent policies under different scenarios, where the set of available models includes:

- M_{const} : a model that always predicts a constant value,
- M_{holy} : a model that predicts the true value during holidays and a constant value otherwise,
- M_{ardw} (AutoRegressive last Day last Week): A statistical model that leverages autoregressive components from the previous day and week for forecasting,
- M_{mlp} (Multi-Layer Perceptron): A neural network model that utilizes multiple layers of interconnected neurons to perform forecasting tasks,
- M_{rf} (Random Forest) : A machine learning model that employs an ensemble of decision trees to enhance the accuracy of forecasts,
- M_{xgb} (eXtreme Gradient Boosting): A gradient boosting framework that combines multiple decision trees to improve forecasting performance.

The subset $M_{ardw}, M_{mlp}, M_{rf}, M_{xgb} \in \mathcal{M}$ contains trainable models taking as input the target and exogenous (if available) time series shifted by one day and one week. This setting allows capturing the trend and seasonality of the time series with minimum parameter overhead.

6.1 Synthetic scenarios

Let $\mathcal{M}$ be the set of available untrained models and $\mathcal{A} = \{\textsc{Select}, \textsc{Keep}, \textsc{Fit}\}$ be the set of actions, where SELECT selects a model, KEEP keeps its current model and FIT refits the model.

Scenario 1 *Let $M_{const}, M_{rf} \in \mathcal{M}$ denote a subset of available forecasting models, and assume that the target time series is cyclic with a frequency that doubles daily. Under these conditions, the optimal forecasting policy is to fit M_{rf} to the data daily, and use it for forecasting.*

Scenario 2 *Let $M_{mlp}, M_{holy} \in \mathcal{M}$ denote a subset of available forecasting models, and assume that the target time series consists of an equal distribution of values, with half occurring during holidays and the other half during normal days. Under these conditions, the optimal forecasting policy is to select M_{holy} for the holiday period and, upon transitioning to a normal day, to switch to M_{mlp} for continued forecasting.*

For each scenario, our agent policy has learned the optimal policy. To validate the learning of the agent during the synthetic scenarios, we further assess its performance by monitoring the MAPE score in a real-world scenario.

6.2 Real-world scenario

Hydro-Québec is a state-owned corporation responsible for the generation, transmission, and distribution of electricity within the province of Québec, Canada. Our agent is evaluated on data provided by Hydro Québec, which consists of hourly electricity load in a remote village of Quebec, Canada for the year 2018.

Initially, all models can be fitted on 11 months of historical data containing the load demand (in MW) and the temperature (in Celsius). A baseline forecast reference is established for a given horizon. The choice of the baseline model is made by selecting the best model in terms of forecast accuracy for the given horizon. During our experimentation, we found M_{mlp} to be the best model having set the horizon to one week. The agent is subsequently trained with the PPO algorithm over 100 episodes. The datetime of the terminal state corresponds to the end of the horizon. Figure 3 illustrates the progression of the learned policy after 10 and 100 episodes.

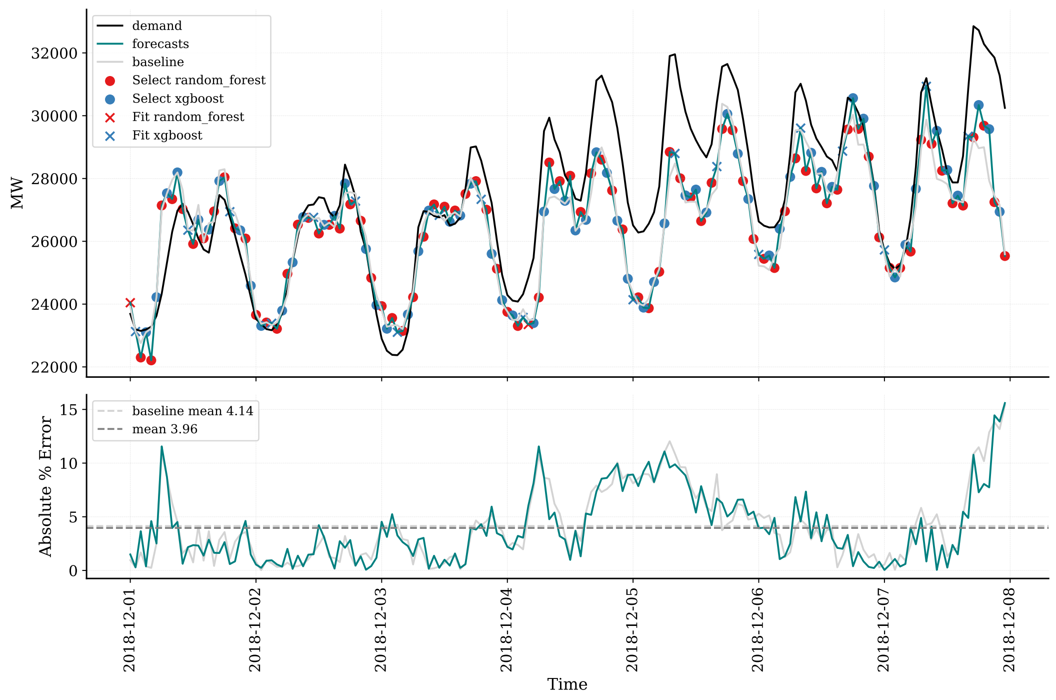
As illustrated in Figure 3, the agent ultimately converges to retaining only the M_{mlp} model, refitting it near turning points, notably during the morning peak where load is harder to forecast and can be interpreted as a good indicator for the upcoming load pattern. Our approach demonstrates an improvement in the baseline MAPE score, reducing it from 4.14% to 3.37%. While the resulting MAPE score remains relatively high, the set of available models prioritizes computational efficiency for faster training. Incorporating additional exogenous time series data or employing more sophisticated forecasting models could further enhance accuracy.

In our experience, we evaluate the performance of the agent model by directly considering the MAPE score function. Although this score is sufficiently good for providing an overall assessment of the model, it outlines a symmetric relationship between the cost of overestimation and underestimation. A better alignment of the forecasts evaluation function with respect to the objective of the grid operator would yield a more adapted policy.

Unlike offline systems, online systems operate in real-time and continuously interact with dynamic data streams. This requirement introduces significant challenges, as resetting the environment in an online system may not always be feasible. In such cases, the methodology could be adapted to incorporate fixed-length rollout windows or truncated episodes to address this limitation. Nonetheless, the REST API framework is well-suited for this application, as it facilitates the management of multiple forecasting models and enables interaction with the environment through a defined set of actions.

7 Conclusion

In this paper, we have introduced a novel approach for managing the lifecycle of forecasting models in a closed-loop manner. Due to the limited understanding of the underlying reasons for model effectiveness, adjustments and tuning during the training phases are often guided by heuristics, such as employing techniques derived from recent empirical studies. Given the inherent complexity of the problem, we have proposed the use of a reinforcement learning agent that autonomously selects the most appropriate model for the current context and retrains models when necessary, employing a reward schema based on MAPE score. We have evaluated our approach on a real-world energy dataset, utilizing a REST API framework to facilitate the deployment and management of models within a closed-loop system, thereby making it suitable for industrial applications. Our approach improves the MAPE score over a baseline model, demonstrating the effectiveness of our method in managing the lifecycle of forecasting models. In future work, we aim to extend the action space of the agent in order to include more advanced actions, such as hyperparameter tuning. Additionally, we plan to extend the set of available models and implement the process within an online system.



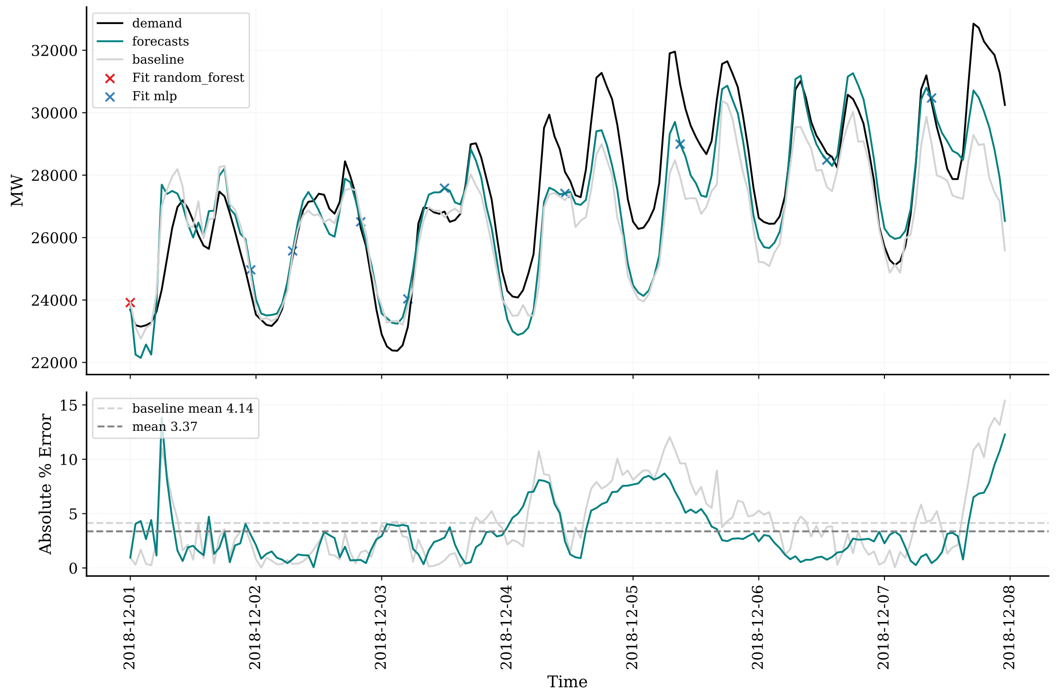


Figure 3: Progression of the learned policy after 10 episodes (top) and 100 episodes (bottom) for a forecasting horizon of 7 days from Saturday December 1st to Friday December 8th of the year 2018.

References

José Ramón Cancelo, Antoni Espasa, and Rosmarie Grafe. Forecasting the electricity load from one day to one week ahead for the spanish system operator. *International Journal of forecasting*, 24 (4):588–602, 2008.

Adam E Clements, AS Hurn, and Zili Li. Forecasting day-ahead electricity load using a multiple equation time series approach. *European Journal of Operational Research*, 251, 2016.

Davide Corradini, Zeno Montolli, Michele Pasqua, and Mariano Ceccato. Deeprest: Automated test case generation for REST APIs exploiting deep reinforcement learning. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1383–1394, 2024.

Jianguang Deng and Panida Jirutitijaroen. Short-term load forecasting using time series analysis: A case study for singapore. In *2010 IEEE conference on cybernetics and intelligent systems*, pp. 231–236. IEEE, 2010.

J. Garcia, J. Kersulis, D. Kosterev, L. Mili, and V. Venkatasubramanian. OpenSTEF: Open source software for transient stability analysis, 2025. URL https://lenergy.org/projects/openstef/. Accessed on 2025-05-15.

RJ Hyndman. *Forecasting: principles and practice*. OTexts, 2018.

Myeongsoo Kim, Saurabh Sinha, and Alessandro Orso. Adaptive REST API testing with reinforcement learning. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 446–458. IEEE, 2023.

Sebastian Kotstein and Christian Decker. Reinforcement learning for IoT interoperability. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pp. 11–18, 2019. DOI: 10.1109/ICSA-C.2019.00010.

Lizhi Liao, Heng Li, Weiyi Shang, and Lei Ma. An empirical study of the impact of hyperparameter tuning and model optimization on the performance properties of deep neural networks. *ACM Transactions on Software Engineering and Methodology*, 31(3):1–40, 2022.

Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37, 2021.

Lakshmi Mandal, Balaji Ganesan, Avirup Saha, and Renuka Sindhgatta. Graph-guided API sequencing with reinforcement learning. In *The Third Learning on Graphs Conference*, 2024.

DQ Mayne. Model predictive control theory and design. *Nob Hill Pub, Llc*, 1999.

Hui Miao, Ang Li, Larry S Davis, and Amol Deshpande. Towards unified data and lifecycle management for deep learning. In *2017 IEEE 33rd International Conference on Data Engineering*, pp. 571–582. IEEE, 2017.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased LSTM: Accelerating recurrent network training for long or event-based sequences. *Advances in neural information processing systems*, 29, 2016.

Maryam Nejati, Nima Amjady, and Hamidreza Zareipour. A new multi-resolution closed-loop wind power forecasting method. *IEEE Transactions on Sustainable Energy*, 2023.

Tien-Quang Nguyen, Nghia-Hieu Cong, Ngoc-Minh Quach, Hieu Dinh Vo, and Son Nguyen. Reinforcement learning-based REST API testing with multi-coverage. *arXiv preprint arXiv:2410.15399*, 2024.

Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2019.

Leonard Richardson and Sam Ruby. *RESTful web services*. "O'Reilly Media, Inc.", 2008.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Lacir J Soares and Marcelo C Medeiros. Modeling and forecasting short-term electricity load: A comparison of methods with an application to brazilian data. *International Journal of Forecasting*, 24, 2008.

Kaleem Ullah, Muhammad Ahsan, Syed Muhammad Hasanat, Muhammad Haris, Hamza Yousaf, Syed Faraz Raza, Ritesh Tandon, Samain Abid, and Zahid Ullah. Short-term load forecasting: A comprehensive review and simulation study with cnn-lstm hybrids approach. *IEEE Access*, 2024.

Jorge Yasuoka, RA Souza, Jose A Jardini, Roberto Castro, Fernando AA Prado, Liliana MV Brittes, Andre LP Cruz, and Hernan P Schmidt. Impact of exogenous variables on estimated values of demand and energy. In *2004 IEEE/PES Transmision and Distribution Conference and Exposition: Latin America (IEEE Cat. No. 04EX956)*, pp. 344–348. IEEE, 2004.

Chi Zhang and Ran Li. A novel closed-loop clustering algorithm for hierarchical load forecasting. *IEEE Transactions on Smart Grid*, 12, 2020.

Yufan Zhang, Qiuwei Wu, Qian Ai, and João PS Catalão. Closed-loop aggregated baseline load estimation using contextual bandit with policy gradient. *IEEE Transactions on Smart Grid*, 13, 2021.

Yufan Zhang, Honglin Wen, Qiuwei Wu, and Qian Ai. Optimal adaptive prediction intervals for electricity load forecasting in distribution systems via reinforcement learning. *IEEE Transactions on Smart Grid*, 2022.

H Zhou, S Zhang, J Peng, S Zhang, J Li, H Xiong, and W Zhang Informer. Beyond efficient transformer for long sequence time-series forecasting. *DOI: https://doi. org/10.1609/aaai. v35i12*, 17325, 2023.