A Project Report
On

DeepComp: Deep reinforcement learning-based renewable energy forecasting

BY

BHAVY GOEL
2019B4A70586P

Under the supervision of
DR. SUMANTA PASARI
DR. RAKHEE

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

MATH F266: STUDY PROJECT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI
(RAJASTHAN)

PILANI CAMPUS

CONTENTS

ABSTRACT	3
INTRODUCTION	4
SYSTEM MODEL	6
METHODOLOGY	8
IMPLEMENTATION	10
EXPERIMENTAL RESULTS	13
CONCLUSION AND FUTURE WORK	25
REFERENCES	26

ABSTRACT

Recently, renewable energy is rapidly integrated into the power grid to prevent climate change, and accurate forecasting of renewable generation becomes critical for reliable power system operation. However, existing forecasting algorithms only focused on reducing forecasting errors without considering error compensability by using a large-scale battery. To incorporate it, a novel strategy called error compensable forecasting is proposed. The objective of forecasting is switched from reducing errors to making errors compensable by leveraging a battery, which in turn reduces the dispatched error, the difference between forecasted value and dispatched value.

The challenging part of the proposed objective lies in that the stored energy at the current time is affected by the previous forecasting result. In this regard, a deep reinforcement learning-based error compensable forecasting framework, called DeepComp, is proposed to have forecasting in the loop of control. This makes an action a continuous forecasted value, which requires a continuous action space. As the action is continuous, proximal policy optimization, which is simple to implement with outstanding performance for continuous control is applied. Extensive experiments with solar and wind power generations show that DeepComp outperforms the conventional forecasting methods by up to 90% and achieves accurate forecasting, e.g., 0.58–1.22% of the mean absolute percentage error.

INTRODUCTION

In the previous years, the use of renewable energy to prevent climate change has increased substantially. The necessity of solar and wind power instead of fossil fuels has been arising. As a result, global penetration of renewable energy rapidly increases, but renewable generation heavily depends on weather conditions such as clouds, temperature, and humidity. This causes substantial uncertainties, and renewable energy forecasting becomes of prime importance.

In the literature, machine learning-based renewable generation forecasting techniques have been recently proposed to improve forecasting accuracy. Specifically, deep learning models have the potential to improve forecasting accuracy by capturing the complex dependencies and uncertainties of data. Advanced Deep Neural Networks(DNNs) showed significantly improved performances compared to conventional machine learning-based schemes even in the very short-term forecasting horizon region. However, forecasting always induces errors. Hence, dispatched error, the difference between the forecasted power and the dispatched power inevitably exists.

Literature considered that the forecasting error is proportional to the dispatched error that is smaller the dispatched error, the smaller the forecasting error. Forecasting algorithms considered reducing forecasting errors by using advanced DNN algorithms. The problem with these is that they consider MSE/MAE as the objective function which does not consider error polarity but simply reduces error distance.

In this regard, a deep reinforcement learning-based error compensable forecasting framework, called DeepComp, is proposed. The objective of forecasting is to make errors compensable now. The challenging part of developing DeepComp lies in that the charging or discharging amount at the current time is determined by the forecasting error resulting from the previous forecasting result. Hence, time-coupling exists between forecasting and battery control, and forecasting should be in the loop of *sequential decision making*. This problem is tackled by leveraging deep reinforcement learning (DRL) which is a combination of classic reinforcement learning and DNN architecture. The inputs and outputs of DNNs become the states and actions in our DRL framework, respectively. This makes an action a continuous forecasted value, which requires a continuous action space. To enable continuous control, we leverage the algorithm called proximal policy optimization (PPO). PPO is based on the actor-critic policy gradient methods and easily solves the trust-region constraint problem that is essential to solving the correlated data problem in DRL. This makes PPO simpler to implement than other DRL algorithms with
outstanding performance.

How does this approach differ from other forecasting algorithms?

1. All subsequent error compensability is considered. This implies, that future impacts, as well as the following outputs, are considered when determining the forecasted power.

2. The currently stored energy is considered as an input of the forecasting algorithm in addition to traditional inputs.
3. The loss function is made by considering error compensability, not simply by the error distance, which aims to reduce the dispatched error, the eventual goal of forecasting.

SYSTEM MODEL

Battery Operation

The energy stored at time slot t, denoted by E_t

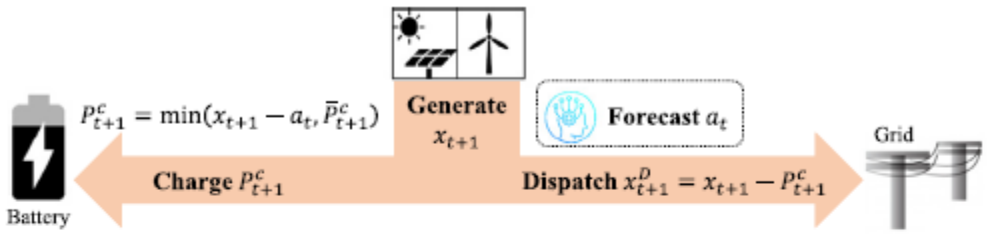
$$E_{\max} \cdot \mathrm{SoC}_{\min} \le E_t \le E_{\max} \cdot \mathrm{SoC}_{\max},$$

Where SoC is the State of Charge of the battery and SoCmin, SoCmax denotes its minimum and maximum respectively.

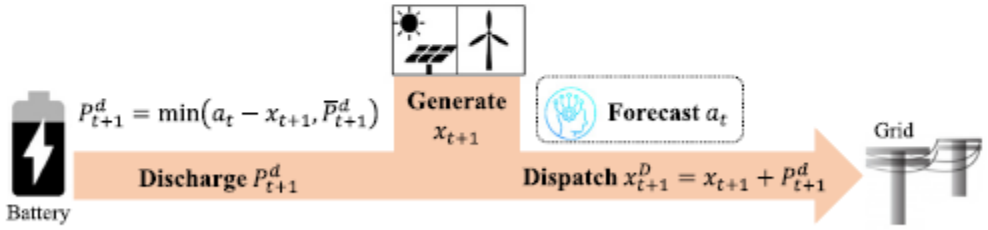
Let P_t^c and P_t^d denote the charging and discharging power limitation at time slot t.

$$\bar{P}_{t+1}^c = \min\left(P_{\max}^c, \quad \frac{1}{\eta_c} \cdot \frac{E_{\max} \cdot \mathrm{SoC}_{\max} - E_t}{\Delta t}\right),$$
$$\bar{P}_{t+1}^d = \min\left(P_{\max}^d, \quad \eta_d \cdot \frac{E_t - E_{\max} \cdot \mathrm{SoC}_{\min}}{\Delta t}\right),$$

where $P_{\max}^c$, $P_{\max}^c$ are the maximum charging power and discharging power of the power conditioning system, Δt is the duration of time slot, ηc, ηd are the charging and discharging efficiencies, respectively.



(a) Battery operation in under-forecasting ($x_{t+1} > a_t$)



(b) Battery operation in over-forecasting ($x_{t+1} < a_t$)

Fig. 1. An overview of the battery operation.

Let x_t be the real generation value in time slot t, and a_t be the forecasted value in the next time slot $t + 1$

$$P^c_{t+1} = \min\left((x_{t+1} - a_t)^+, \bar{P}^c_{t+1}\right),$$
$$P^d_{t+1} = \min\left((a_t - x_{t+1})^+, \bar{P}^d_{t+1}\right),$$

where $(z)^+ = \max(z, 0)$. Accordingly, E_t evolves in time as follows

$$E_{t+1} = E_t + \eta_c P^c_{t+1}\Delta t - \frac{1}{\eta_d} P^d_{t+1}\Delta t,$$

and the dispatched power to the grid at the next time slot t+1, denoted by x^D_{t+1}, is defined as

$$x^D_{t+1} = x_{t+1} - P^c_{t+1} + P^d_{t+1}.$$

The dispatched error at the next time slot t + 1, denoted by e^D_{t+1}, is defined as

$$e^D_{t+1} = a_t - x^D_{t+1}.$$

METHODOLOGY

P1: Error Compensable Forecasting Problem

$$\min_{\{a_t\}_{t=0}^{\infty}} \quad \mathbb{E}_{\{x_{t+1}\}_{t=0}^{\infty}} \left[\sum_{t=0}^{\infty} \gamma^t f_{t+1}^{D} \middle| E_0 = E_{\text{init}} \right],$$

where E_{init} is the initially stored energy and $\gamma \in (0, 1)$ is a discount factor that determines the importance of future impacts.

We tackle P1 by leveraging RL that has interaction between an agent and environment.

In the RL framework, there is a set of states and a set of actions. At time slot t, the agent takes an action $a_t \in$ A state $s_t \in$ and goes to the next state $s_{t+1} \in$ S with a reward r_{t+1}.
RL is different from conventional machine learning as it aims to maximize all the discounted subsequent rewards, not the immediate one.

State value function $V^{\pi}(s)$, the state–action-value function $Q^{\pi}(s, a)$, and the advantage function $A^{\pi}(s, a)$ as follow:

$$V^{\pi}(s) = \mathbb{E}_{\tau\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| s_0 = s \right],$$

$$Q^{\pi}(s, a) = \mathbb{E}_{\tau\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| s_0 = s, a_0 = a \right],$$

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s),$$

where the trajectories $\tau\pi = (s0, a0, r1, s1, \ldots)$ are drawn from the trajectory distribution induced by the policy π.

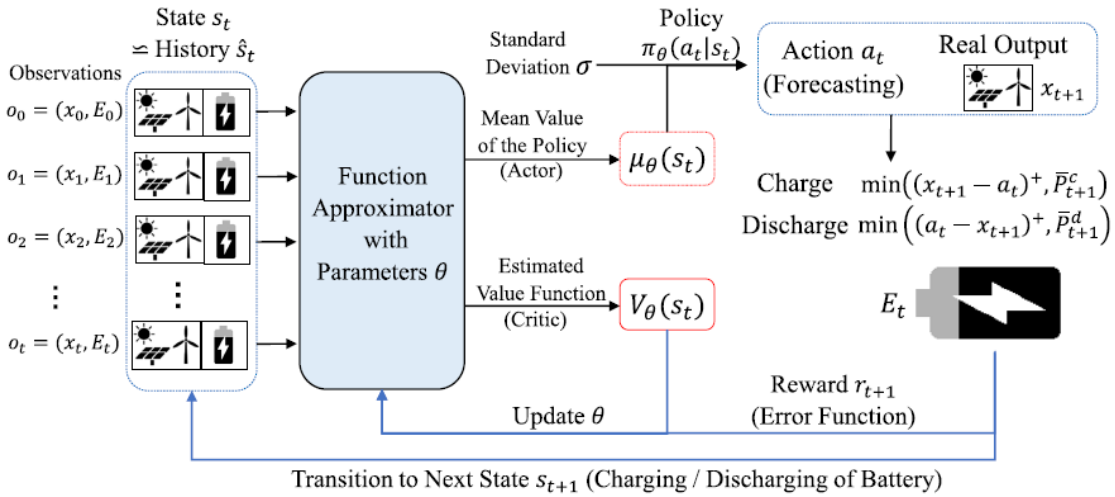


Fig. 3. A framework of DeepComp.

P2: Error Compensable Forecasting Problem with Policy-based RL

$$\max_{\theta} \quad \mathbb{E}_{\{x_{t+1},a_t\}_{t=0}^{\infty}} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| E_0 = E_{\text{init}} \right],$$

where $a_t \sim \pi_\theta(\cdot|s_t)$, $\forall t$. As r_{t+1} and E_{t+1} are determined by x_{t+1}, E_t, and a_t, the distribution of $\{x_{t+1}, a_t\}_{t=0}^{\infty}$ is equivalent to the distribution of $\tau^{\pi\theta}$.

The above equation can be rewritten as

$$\mathbb{E}_{\{x_{t+1},a_t\}_{t=0}^{\infty}} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| E_0 = E_{\text{init}} \right]$$
$$= \mathbb{E}_{s\sim\rho_0,\tau^{\pi\theta}} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \middle| s_0 = s \right] = \mathbb{E}_{s\sim\rho_0} \left[V^{\pi_\theta}(s) \right],$$

where $\varrho_0(s) = P(s_0 = s)$

P3: Error Compensable Forecasting Problem with Trust Region Constraints

$$\max_{\theta} \quad \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}},a\sim\pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} Q^{\pi_{\theta_{\text{old}}}}(s,a) \right],$$
$$\text{s.t.} \quad \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}} \left[D_{\text{KL}} \left(\pi_{\theta_{\text{old}}}(\cdot|s) \;\|\; \pi_\theta(\cdot|s) \right) \right] \le \epsilon.$$

We apply PPO, which is known to be much simpler to implement with outstanding performance. PPO uses the clipped surrogate objective

$$\hat{J}_\pi^{\text{clip}}(\theta) \triangleq \hat{\mathbb{E}}_t \left[\min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t,\; g(\epsilon, \hat{A}_t) \right) \right],$$

where A_t is an estimate of advantage function in PPO, and

$$g(\epsilon, \hat{A}_t) = \begin{cases} (1+\epsilon)\hat{A}_t & \text{if } \hat{A}_t \ge 0, \\ (1-\epsilon)\hat{A}_t & \text{if } \hat{A}_t < 0, \end{cases}$$

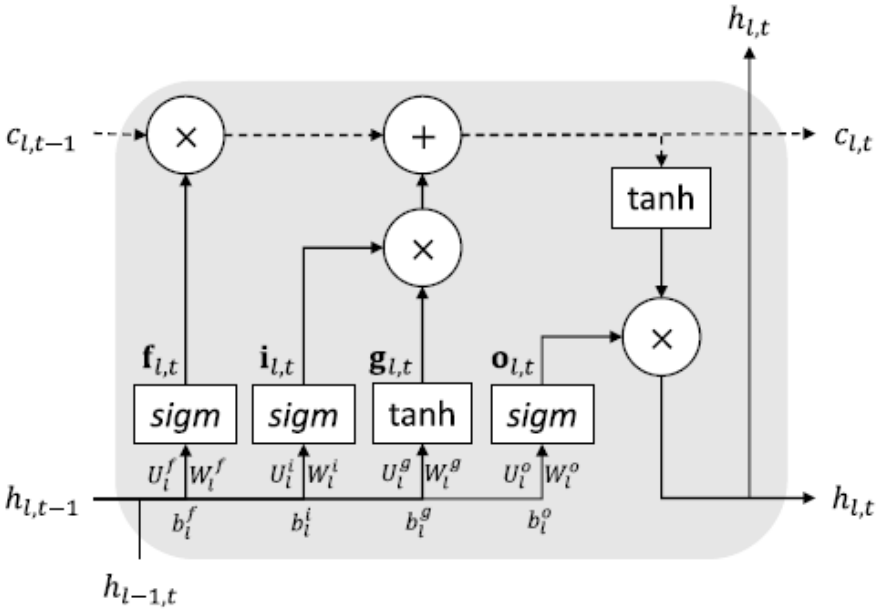
where $\lambda \in (0, 1)$ is a controlled parameter for the bias and variance tradeoff

$$\mathcal{L}(\theta) = -\hat{J}_\pi^{\text{clip}}(\theta) + C\,\mathcal{L}_V(\theta),$$

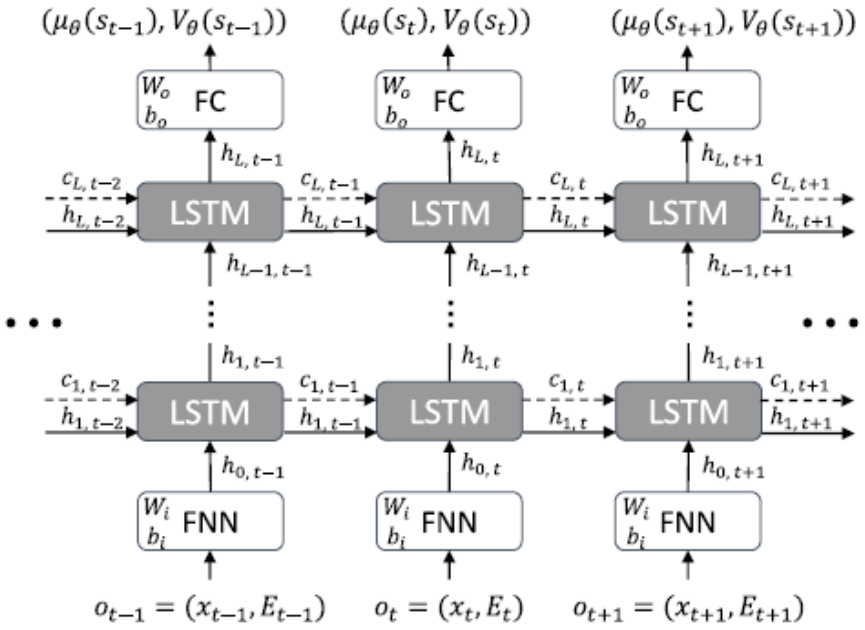
where C is a coefficient of the value function learning.

IMPLEMENTATION

Deep Learning Architecture



(a) LSTM cell



(b) LSTM structure

Fig. 4. The proposed LSTM structure for error compensable forecasting.

We consider RNN for time-series forecasting. RNN shows better performance with time-series history in a partially observable environment than using FNN. LSTM networks have been used that have better learning performance compared to the basic RNN by resolving the vanishing gradient problem. LSTM networks are also widely used to forecast renewable energy generation and show better performance than FNN.

The input vectors o_t = (x_t,E_t) pass through the FNN layer first, and its output becomes the first input of the LSTM, $h_{0,t}$, given by

$$h_{0,t} = \mathrm{ReLU}\left(W_i o_t + b_i\right),$$

where W_i and b_i are weight matrix and bias vector in the input FNN layer, and ReLU is rectified linear unit defined as ReLU(z) = max(z, 0).
Other relations are described as

$$\mathbf{f}_{l,t} = sigm(U_l^f h_{l,t-1} + W_l^f h_{l-1,t} + b_l^f),$$

$$\mathbf{i}_{l,t} = sigm(U_l^i h_{l,t-1} + W_l^i h_{l-1,t} + b_l^i),$$

$$\mathbf{g}_{l,t} = \tanh(U_l^g h_{l,t-1} + W_l^g h_{l-1,t} + b_l^g),$$

$$\mathbf{o}_{l,t} = sigm(U_l^o h_{l,t-1} + W_l^o h_{l-1,t} + b_l^o),$$

$$c_{l,t} = \mathbf{f}_{l,t} \odot c_{l,t-1} + \mathbf{i}_{l,t} \odot \mathbf{g}_{l,t},$$

$$h_{l,t} = \mathbf{o}_{l,t} \odot \tanh\left(c_{l,t}\right),$$

where U_l^f, W_l^f, U_l^i, W_l^i, U_l^g, W_l^g, U_l^o, and W_l^o are weight matrices in $l$thlayer, b_l^f, b_l^i, b_l^g and b_l^o are bias vectors in lth layer, $sigm$ is the sigmoid activation function, and $\odot$ denotes the Hadamard product

The outputs of actor $\mu_\theta(s_t)$ and critic $V_\theta(s_t)$

$$\left(\mu_\theta(s_t), V_\theta(s_t)\right) = W_o h_{L,t} + b_o,$$

where W_o and b_o are weight matrix and bias vector in the output layer, respectively.

Algorithm 1: Proposed DeepComp Algorithm

Initialize the LSTM network parameters θ;
Initialize the initial cell state vectors and hidden output vectors $\{c_{l,-1}, h_{l,-1}\}_{l=1}^{l=L}$ to all-zero vector;
Receive initial observation $o_0 = (x_0, E_0)$;
repeat
  Set initial state $s_0 \leftarrow \left(\{c_{l,-1}, h_{l,-1}\}_{l=1}^{l=L}, o_0\right)$;
  for $t \leftarrow 0$ **to** $T-1$ **do**
    Output $\left(\mu_\theta(s_t), V_\theta(s_t)\right)$ and $\{c_{l,t}, h_{l,t}\}_{l=1}^{l=L}$;
    Execute action a_t according to the $\mu_\theta(s_t)$;
    Receive x_{t+1};
    Calculate reward $r_{t+1} = -f_{t+1}^{D}$ via (2)–(7);
    Calculate E_{t+1} via (2)–(4);
    Set next observation $o_{t+1} \leftarrow (x_{t+1}, E_{t+1})$;
    Set next state $s_{t+1} \leftarrow \left(\{c_{l,t}, h_{l,t}\}_{l=1}^{l=L}, o_{t+1}\right)$;
  end
  Output $V_\theta(s_T)$;
  $\theta_{\text{old}} \leftarrow \theta$;
  Update $\mathcal{L}(\theta)$ for K epochs with learning rate α;
  $\{c_{l,-1}, h_{l,-1}\}_{l=1}^{l=L} \leftarrow \{c_{l,T-1}, h_{l,T-1}\}_{l=1}^{l=L}$;
  $o_0 \leftarrow o_T$
until *convergence*;

EXPERIMENTAL RESULTS

Data description

We used the 60-min time series real-world open datasets for GHI and WIND SPEED from January 1st, 2011 to December 31st, 2014 provided by NSRDB. The forecasting horizon of the very short-term forecasting is up to one hour, and the bids are generally made in an hourly manner. If training data have large values, the weights of neural networks can be very small, which makes it hard to train the DNN. So, we normalize the data by the generation capacity and denormalize the forecasting result to obtain valid outputs. In solar power datasets, the data during the night (zero-value data) is excluded.
To generalize the proposed DRL model, we isolate the datasets into training datasets, validation datasets, and test datasets as in conventional supervised learning, and it is important to make the three datasets have the environments of the same characteristics. In this regard, each dataset has at least one year to validate the performance of all seasons. Also, one to two years long datasets are generally used for training the DNN for renewable energy forecasting. Therefore, the first two years' data are used for training, the following one-year data are used for validation, and the last one year is used for tests. It should be noted that the ratio of training, validation, and test sets is normally, 7:1.5:1.5, or 6:2:2. Hence, our evaluation uses more validation and test sets, which implies better generalization.

Battery Model

For initializing phase, $E_0 = 0.5 \times E_{max}$ is set, i.e. half stored energy. Also, $\beta c = \beta d = 0.01$, which is much smaller than 1, to make $e^D_{t+1} = 0$ as the primary objective and discourage the use of battery when $e^D_{t+1} = 0$.
Other parameters are as follows:

Battery related parameters.

Parameters	Δt	η_c	η_d	SoC_{min}	SoC_{max}	P^c_{max}/E_{max}	P^d_{max}/E_{max}	β_c	β_d
Value	1 h	0.9	0.9	0.1	0.9	1/3	1/3	0.01	0.01

Comparison methods

1. Battery-no-use: At first, the conventional forecasting method is evaluated, and the performances are evaluated without using a battery. The focus is on LSTM-based architecture as it shows the best performance in renewable energy forecasting problems. LSTM is trained with the MSE with the real generation data, which is defined as $(\theta) = \hat{E}_t [(x_{t+1} - \mu_\theta (s_t))^2]$ for every batch. The node of stored energy E_t from the input layer and the node of estimated value function $V_\theta(s_t)$ from the output layer are removed since they

are not required in the conventional forecasting method. The forecasted values are determined as $a_t = \mu_\theta(s_t),\ \forall t$.

2. Naive error compensation (NEC): The forecasted value a_t is the same as the Battery-no-use case. However, in each time slot, the errors are naively compensated as much as possible without concerning the future.
3. Error compensation control (ECC): Unlike NEC, it determines the charging or discharging power of the battery concerning the future error as well. Thus the charging power, denoted by q_t, becomes the actions of the RL. The action q_t is determined by the expected SARSA. The forecasted value a_t is the same as the Battery-no-use case, but the real generation x_{t+1} and the stored energy E_t are changed by q_t as follows:

$$x_{t+1} \leftarrow x_{t+1} - q_t\ ,$$
$$E_t \leftarrow \begin{cases} E_t + \eta_c q_t\ , & \text{if } q_t \geq 0, \\ Et + q_t/\eta_d\ , & \text{if } q_t < 0. \end{cases}$$

4. ECC+: This is an advanced version of ECC. The overall process is the same with the ECC case except that the P^c_{t+1} and P^d_{t+1} are not 0 and are set by

$$\bar{P}^c_{t+1} = \min\left(P^c_{\max},\ \ \frac{1}{\eta_c}\cdot\frac{E_{\max}\cdot \mathrm{SoC}_{\max} - E_t}{\Delta t}\right),$$
$$\bar{P}^d_{t+1} = \min\left(P^d_{\max},\ \ \eta_d\cdot\frac{E_t - E_{\max}\cdot \mathrm{SoC}_{\min}}{\Delta t}\right),$$

That is, for each time, the remaining errors are additionally compensated by the amount that can be compensated from the battery.

Hyperparameter selection

To determine the hyperparameters of each model, the mean absolute percentage error (MAPE) of the dispatched error is observed over a validation set,

$$\mathrm{MAPE} = \frac{100}{|\mathcal{V}|}\sum_{t\in\mathcal{V}}\left|\frac{e^D_{t+1}}{x^D_{t+1}}\right| [\%].$$

All networks are trained by the Adam optimizer with the learning rate α = 0.001 and the mini-batch size T = 128. The framework is built using Pytorch.

The first critical hyperparameter is ϵ, which is a key hyperparameter for PPO methods. The selected ϵ is 0.01 for the solar case and 0.1 for the wind case.

The second critical hyperparameter is the discount factor γ to tradeoff the immediate reward and future reward. Under γ = 0.99, the proposed method shows the best final performance.

The third critical hyperparameter is the standard deviation σ to tradeoff exploration and exploitation. The final performance is similar for the case of σ = 0.05 and σ = 0.1, but

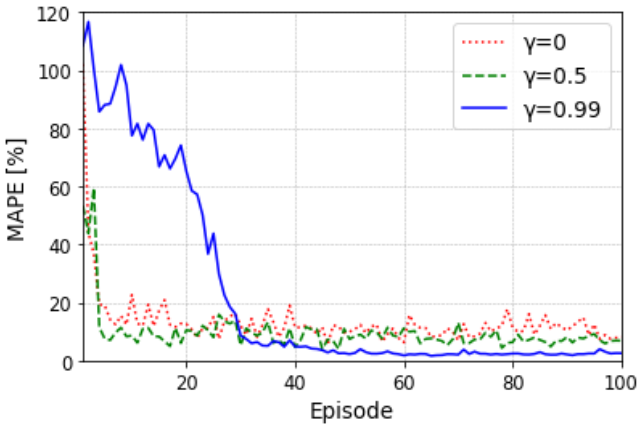
convergence becomes much faster when σ = 0.1 because exploitation is too much prioritized under σ = 0.05.

PPO related hyperparameters.

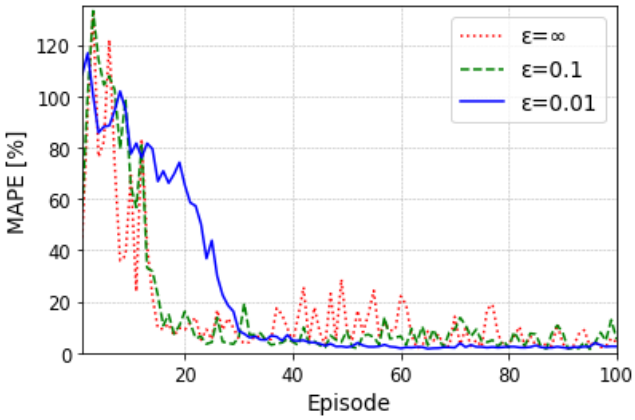
Parameters	T	α	K	γ	λ	ϵ	C	σ
Value	128	0.001	3	0.99	0.95	0.01 (solar), 0.1 (wind)	1	0.1

Performance evaluation

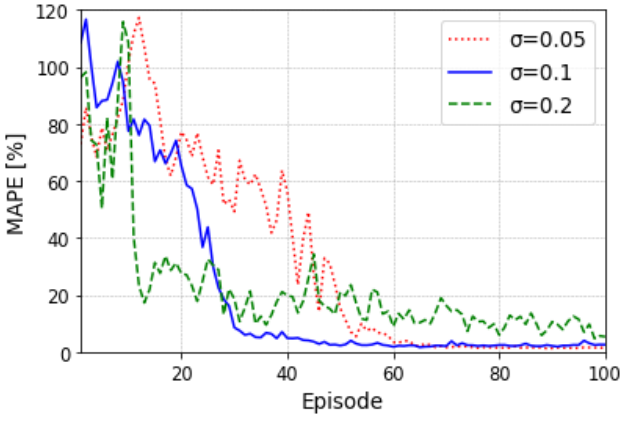
Solar dataset. E_{max} = 0.4 p.u.



Discount factor

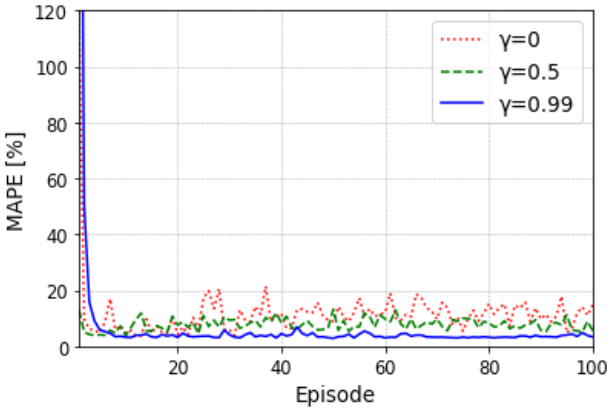


Clipping

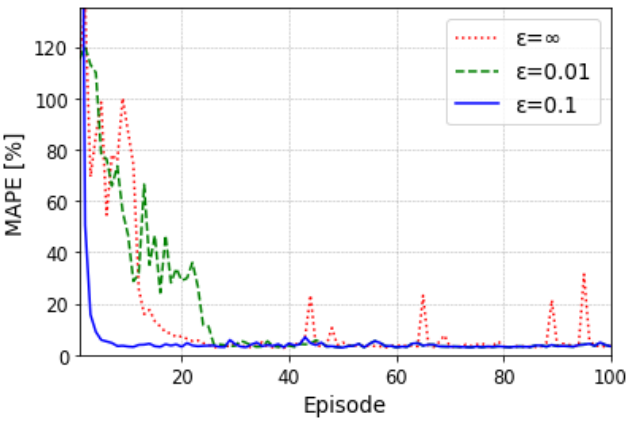


Standard Deviation

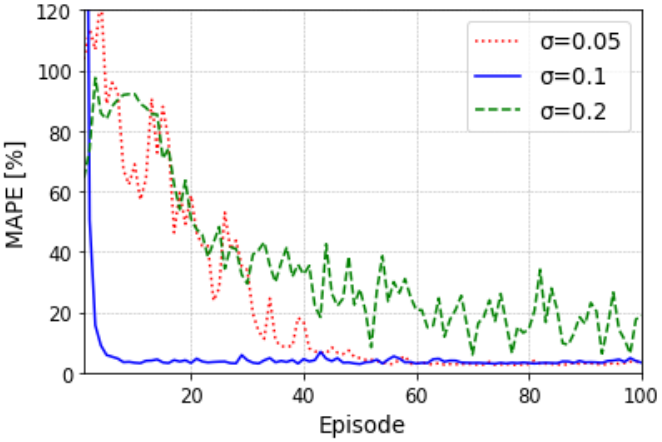
Wind dataset. E_{max} = 0.3 p.u.



Discount factor



Clipping



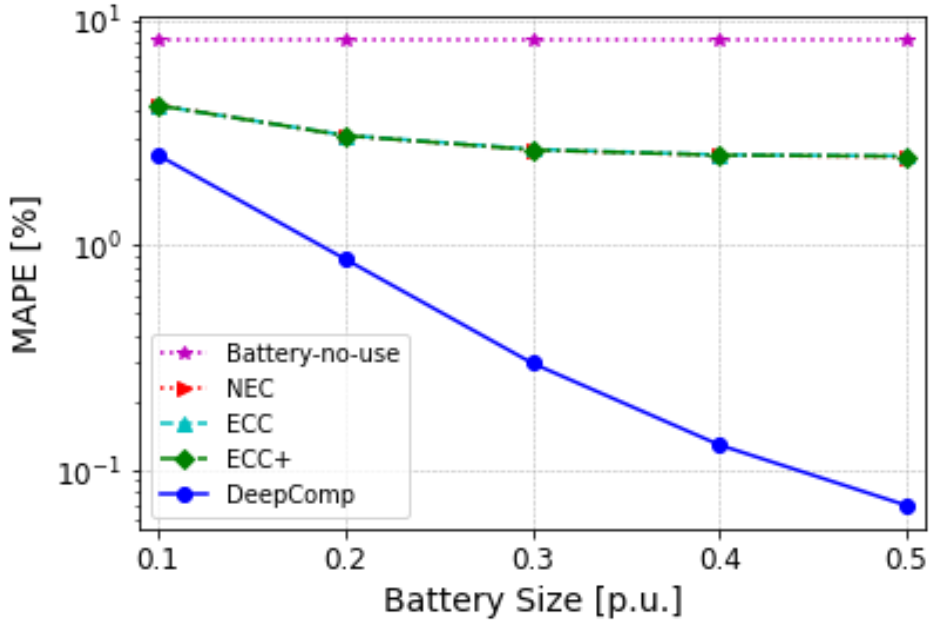
Standard deviation

Wind Dataset:

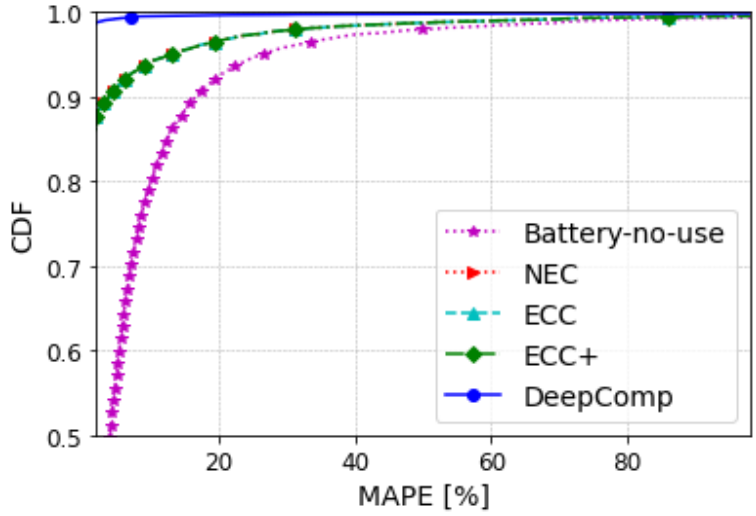
E_{max}

	0.1 p.u.	0.2 p.u.	0.3 p.u.	0.4 p.u.	0.5 p.u.
MAPE: Battery-no-use	8.160	8.160	8.160	8.160	8.160
MAPE: NEC	4.210	3.080	2.660	2.520	2.480
MAPE: ECC	4.190	3.090	2.680	2.530	2.500
MAPE: ECC+	4.190	3.080	2.660	2.520	2.490
MAPE: DeepComp	2.520	0.870	0.300	0.130	0.070

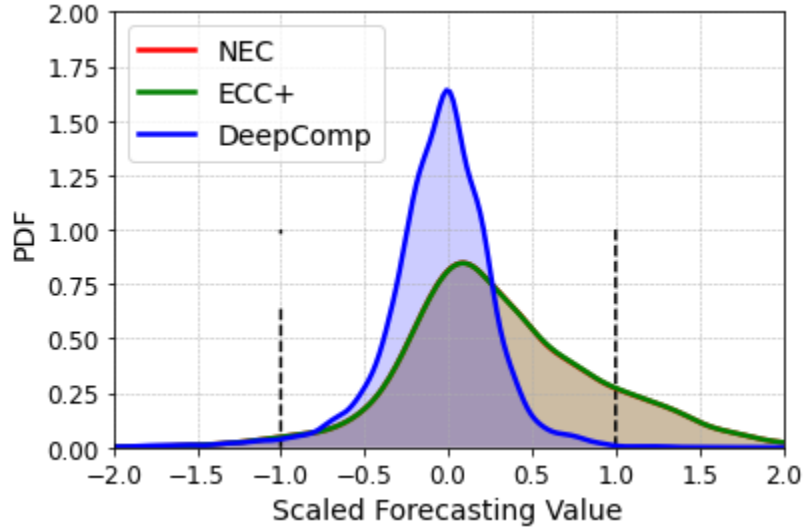
Experiment Results

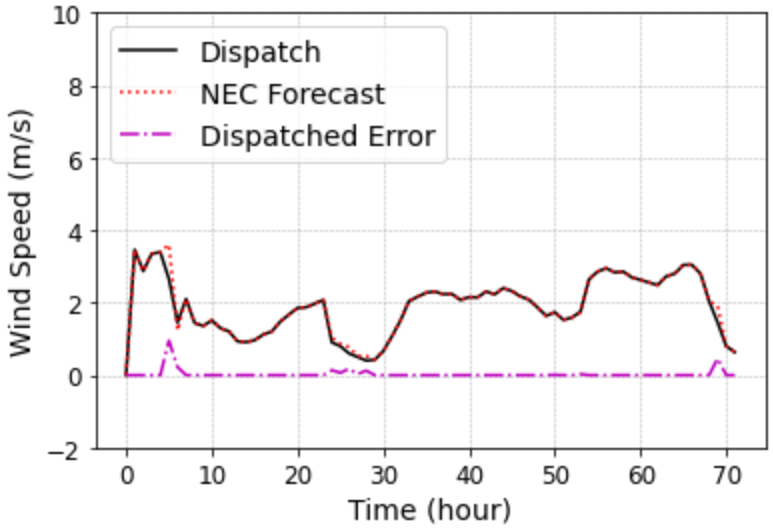


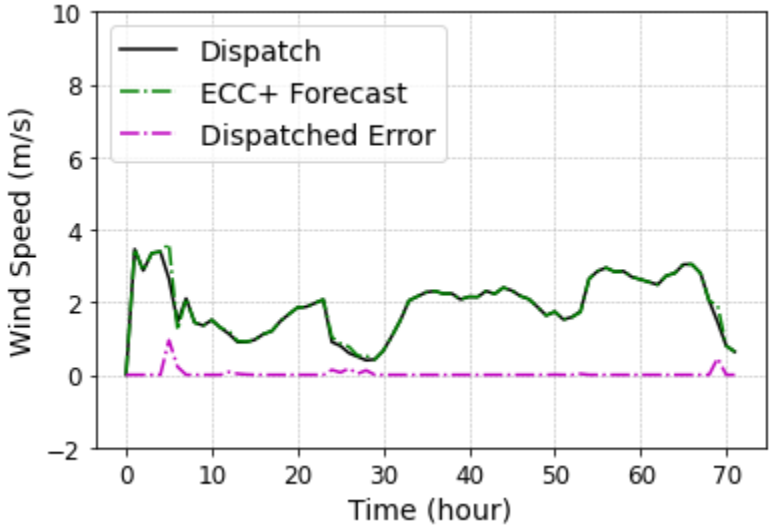
CDF of the MAPE(E_{max} = 0.3)

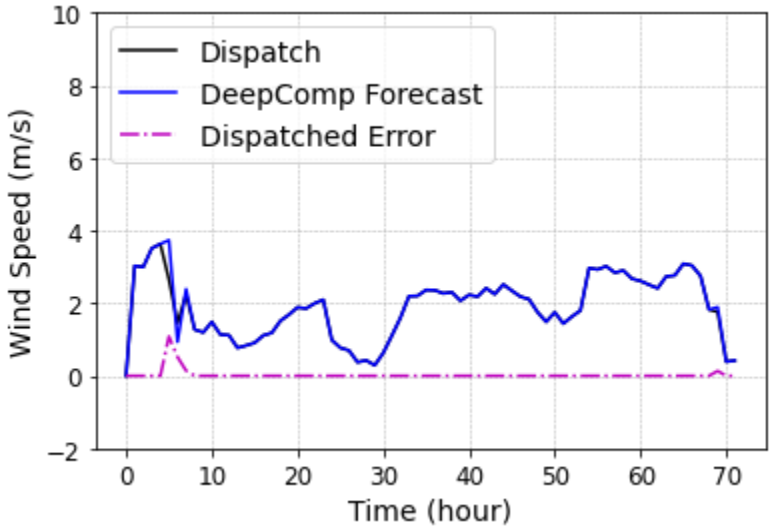


PDF of the scaled forecasting value(E_{max} = 0.3)

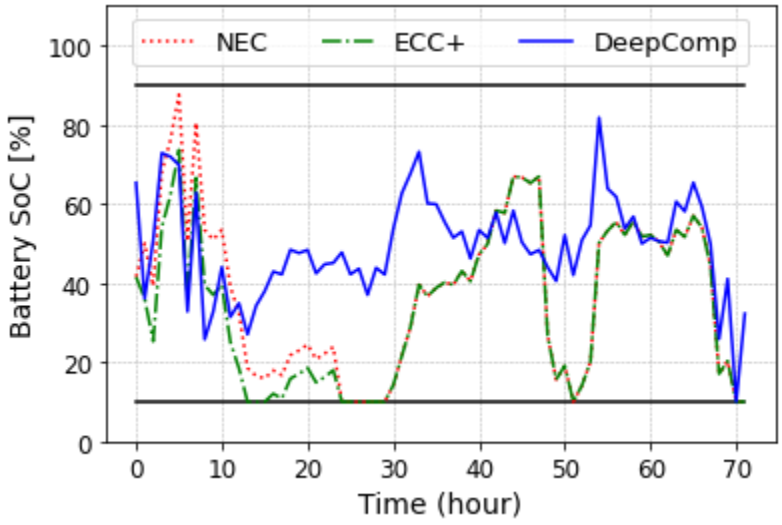



Dispatch
NEC Forecast
Dispatched Error
Wind Speed (m/s)
Time (hour)

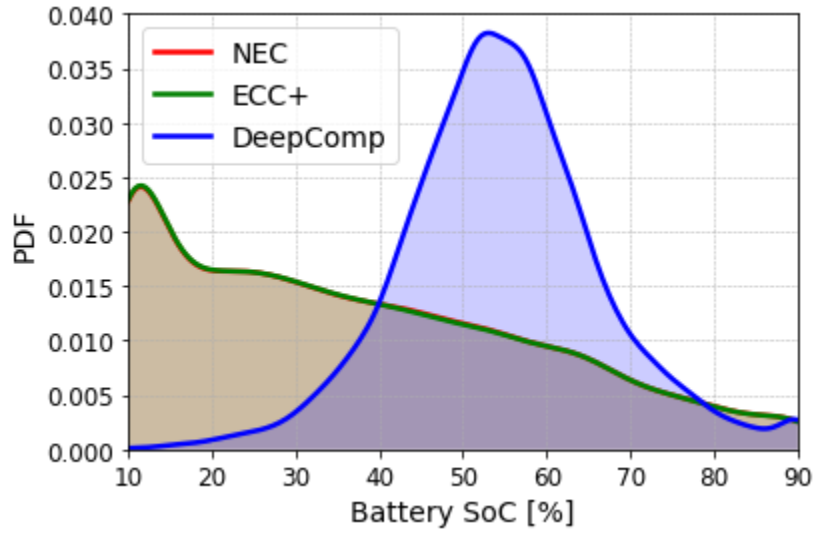

Dispatch
ECC+ Forecast
Dispatched Error
Wind Speed (m/s)
Time (hour)


Dispatch
DeepComp Forecast
Dispatched Error
Wind Speed (m/s)
Time (hour)

Charging/Discharging Patterns(E_{max} = 0.3)



PDF of the battery SoC(E_{max} = 0.3)

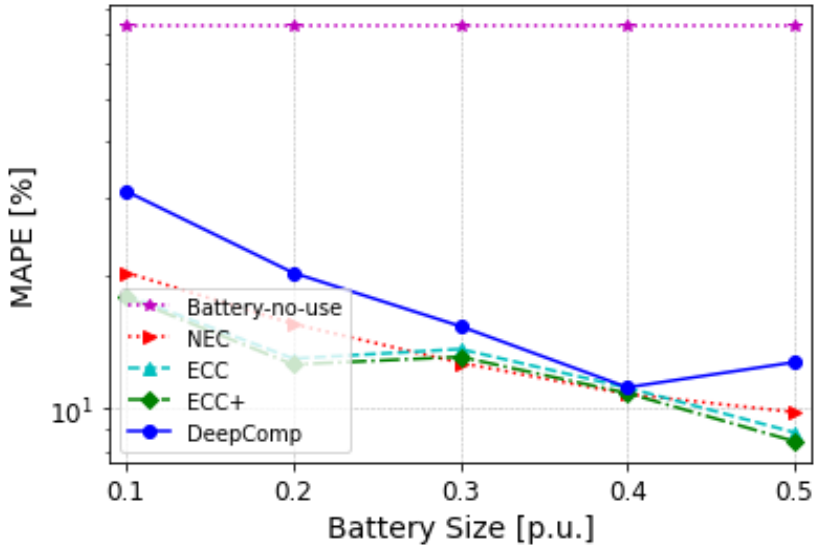


Solar Dataset:

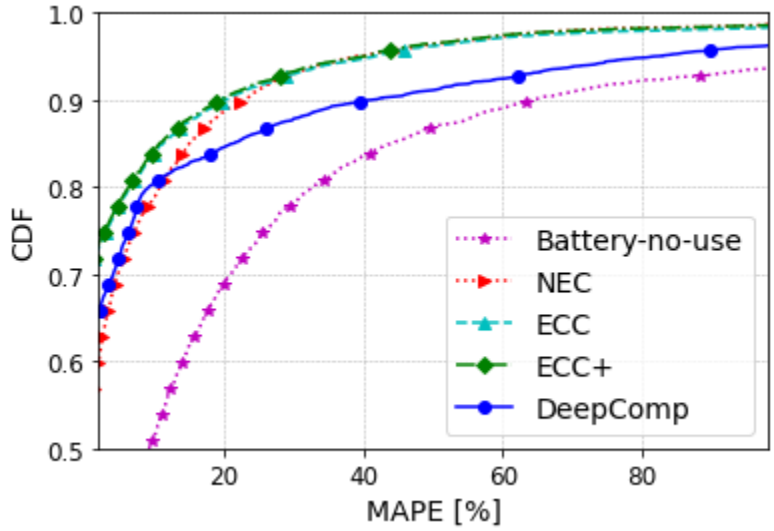
E_{max}

	0.1 p.u.	0.2 p.u.	0.3 p.u.	0.4 p.u.	0.5 p.u.
MAPE: Battery-no-use	72.980	72.980	72.980	72.980	72.980
MAPE: NEC	20.290	15.530	12.690	10.790	9.820
MAPE: ECC	18.040	12.970	13.630	11.150	8.840
MAPE: ECC+	17.840	12.580	13.090	10.840	8.440
MAPE: DeepComp	30.960	20.260	15.350	11.170	12.730

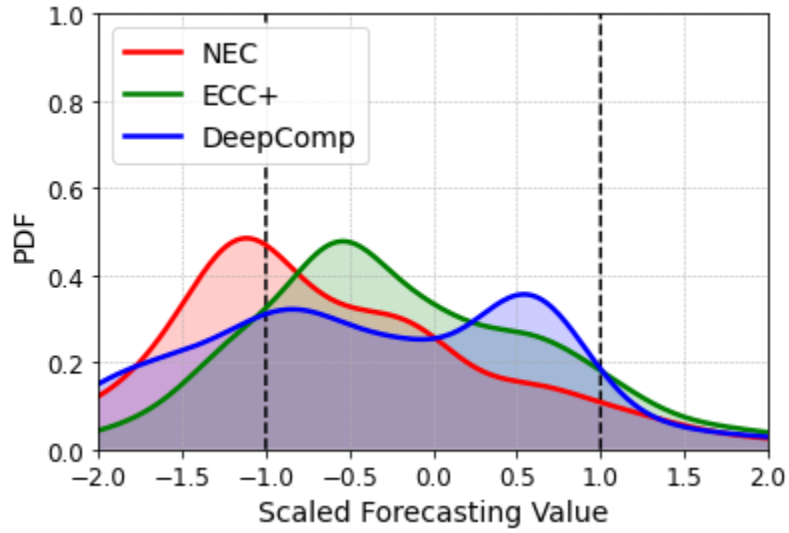
Experiment Results



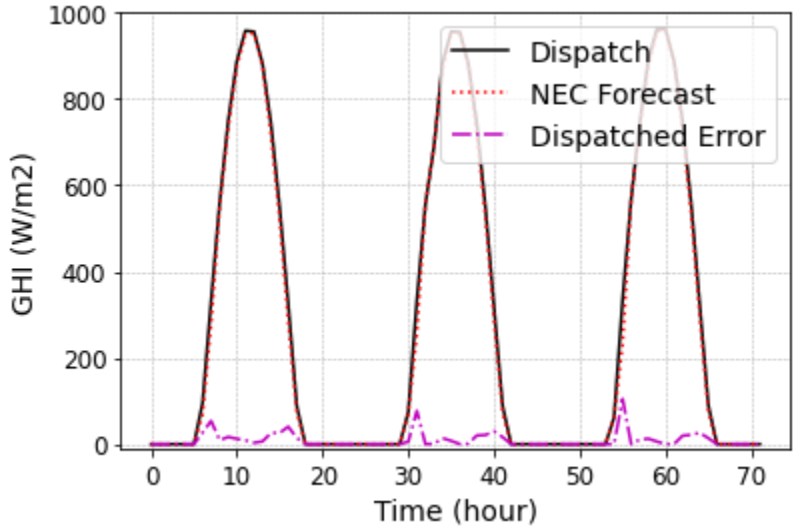
CDF of the MAPE(E_{max} = 0.4)

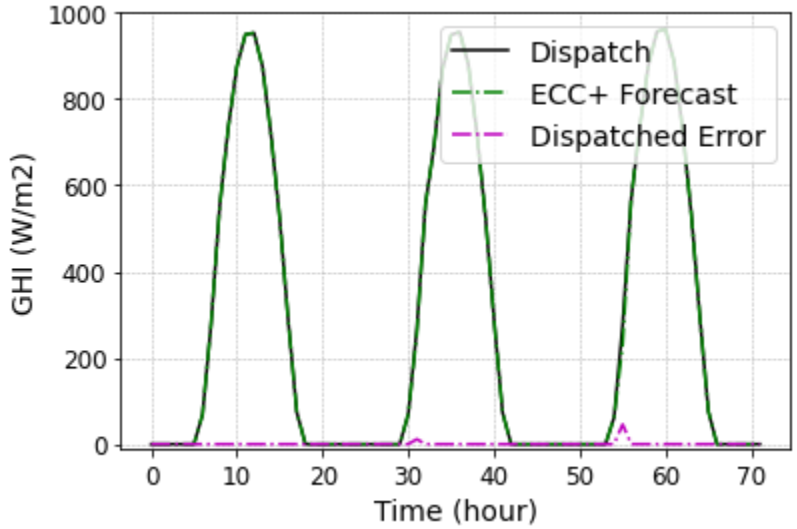


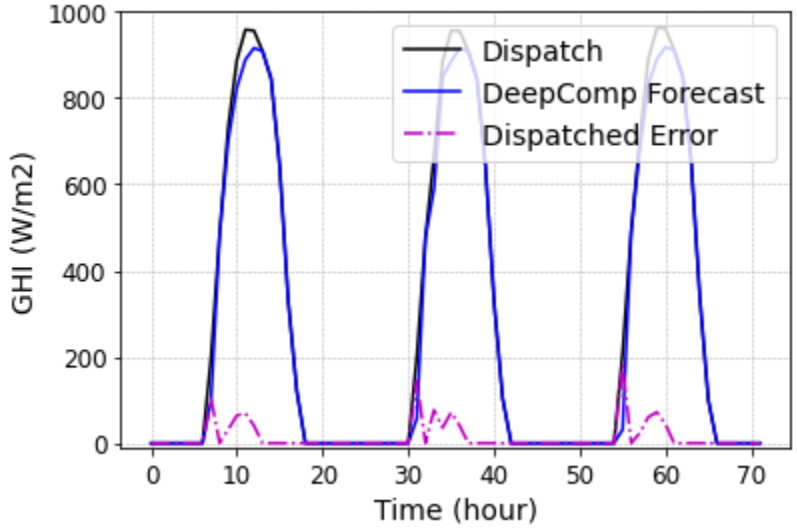
PDF of the scaled forecasting value(E_{max} = 0.4)



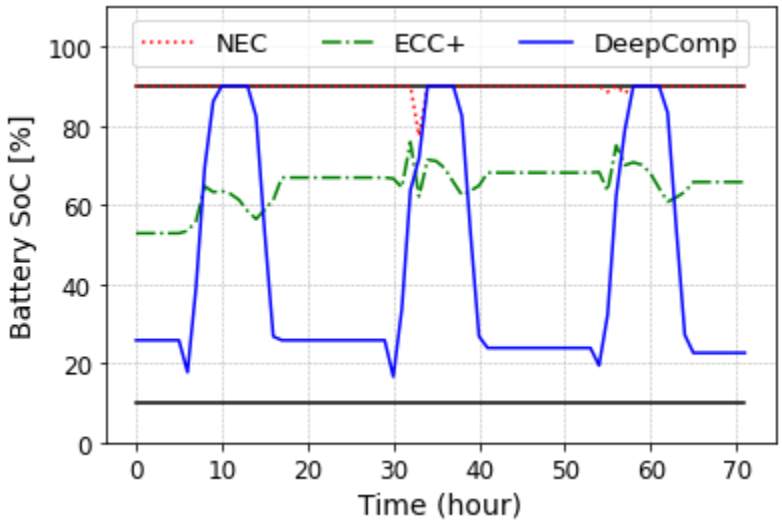
Dispatched solar power and forecasting over 3 days(E_{max} = 0.4)



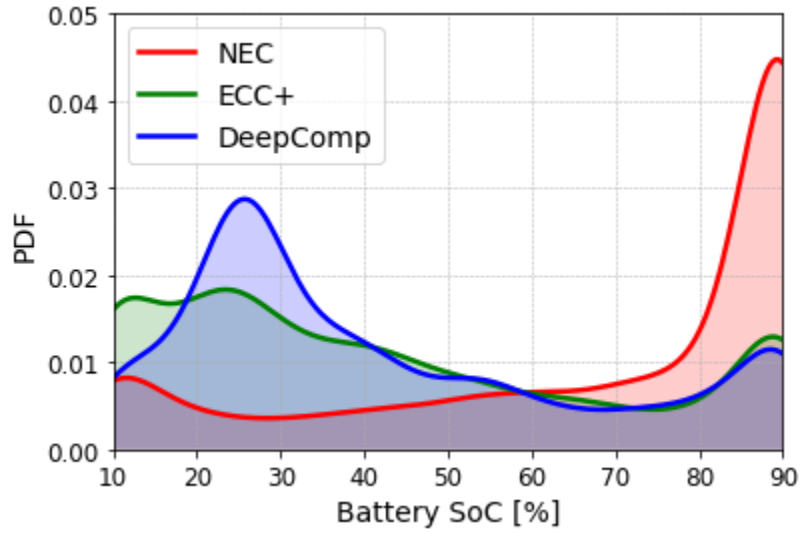




Charging/Discharging Patterns(E_{max} = 0.4)



PDF of the battery SoC(E_{max} = 0.4)



CONCLUSION AND FUTURE RESEARCH

A novel forecasting strategy called DeepComp for renewable energy is proposed where the objective is switched from reducing errors to making errors compensable by using the battery. The loss function in our model aims to reduce the dispatched error, not simply reduce the distance between forecasting and real values. Since the stored energy is affected by the previous forecasting result, we tackle the problem by leveraging DRL having forecasting in the loop of control. Specifically, the RL framework is combined with the DNN architecture of traditional forecasting; the inputs and outputs of DNNs become the states and actions in our DRL framework. As the action is a continuous forecasted value, we apply PPO that is simple to implement with outstanding performance for continuous control. The proposed framework can be easily combined with new forecasting algorithms to come because it does not require changing the model structure. The proposed DeepComp shows significantly better performance than the other forecasting strategies for both solar power and wind power cases, e.g., the MAPE is reduced by about 90%.

The proposed model showed outstanding performance for the wind dataset but the same was not observed for the solar dataset where the model lagged other traditional models.

Future research can be extended in two possible directions. One is to generalize our one-step ahead forecasting into a multi-step ahead (such as day ahead) short-term forecasting algorithm. It requires multiple action RL as we need to output the forecasted value for each time slot. The other is to examine the economic impact such as bidding profit in renewable energy markets.

GITHUB LINK: Bhavy0906/DeepCom (github.com)

REFERENCES

1. Mashlakov A, Kuronen T, Lensu L, Kaarna A, Honkapuro S. Assessing the performance of deep learning models for multivariate probabilistic energy forecasting. Appl Energy 2021;285:116405.
2. Zheng J, Zhang H, Dai Y, Wang B, Zheng T, Liao Q, et al. Time series prediction for output of multi-region solar power plants. Appl Energy 2020;257:114001.
3. Kim JH, Powell WB. Optimal energy commitments with storage and intermittent supply. Oper Res 2011;59(6):1347–60.
4. Ryu S, Bae S, Lee J-U, Kim H. Gaussian residual bidding based coalition for two-settlement renewable energy market. IEEE Access 2018;6:43029–38.
5. Ryu S, Noh J, Kim H. Deep neural network based demand side short term load forecasting. Energies 2017;10(1):3.
6. Zhang C, Vinyals O, Munos R, Bengio S. A study on overfitting in deep reinforcement learning. 2018, arXiv preprint.
7. Bae KY, Jang HS, Sung DK. Hourly solar irradiance prediction based on support vector machine and its error analysis. IEEE Trans Power Syst 2017;32(2):935–45.
8. Bae KY, Jang HS, Jung BC, Sung DK. Effect of prediction error of machine learning schemes on photovoltaic power trading based on energy storage systems. Energies 2019;12(7):1249.
9. Oh E, Wang H. Reinforcement-learning-based energy storage system operation strategies to manage wind power forecast uncertainty. IEEE Access 2020;8:20965–76.
10. Oh E. Reinforcement-learning-based virtual energy storage system operation strategy for wind power forecast uncertainty management. Appl Sci 2020;10(18):6420.
11. NSRDB Data Viewer (nrel.gov)
12. DeepComp: Deep reinforcement learning based renewable energy error compensable forecasting