에너지 AI와 강화학습

Hongseok Kim

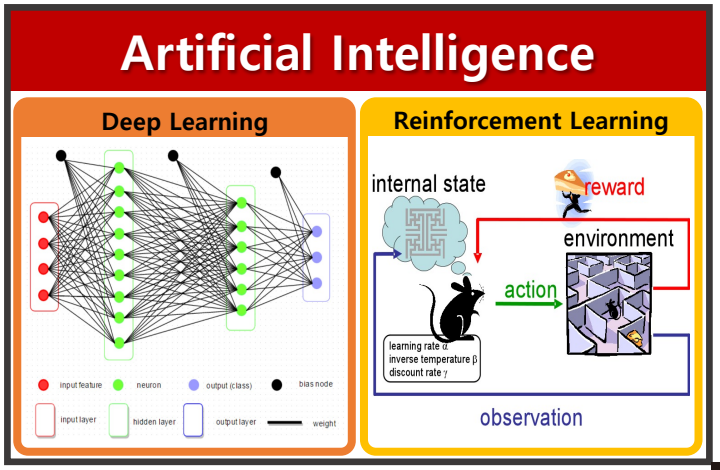
2022. 3

Sogang University, EE

NICELAB
Networking for Intelligence Communications and Energy

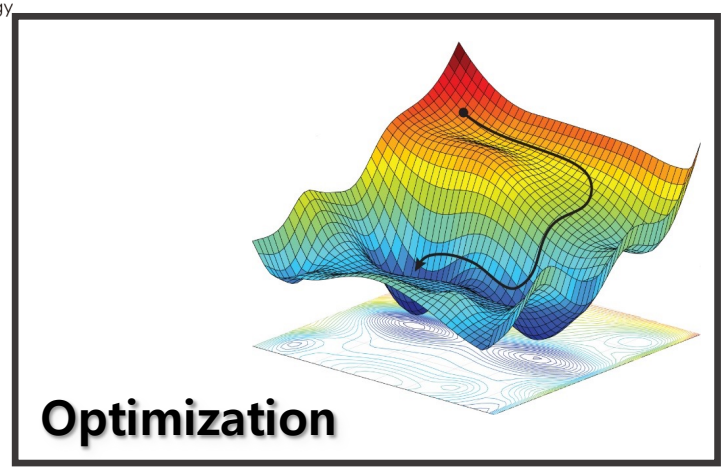
서강대학교 SOGANG UNIVERSITY

Energy ICT Domain

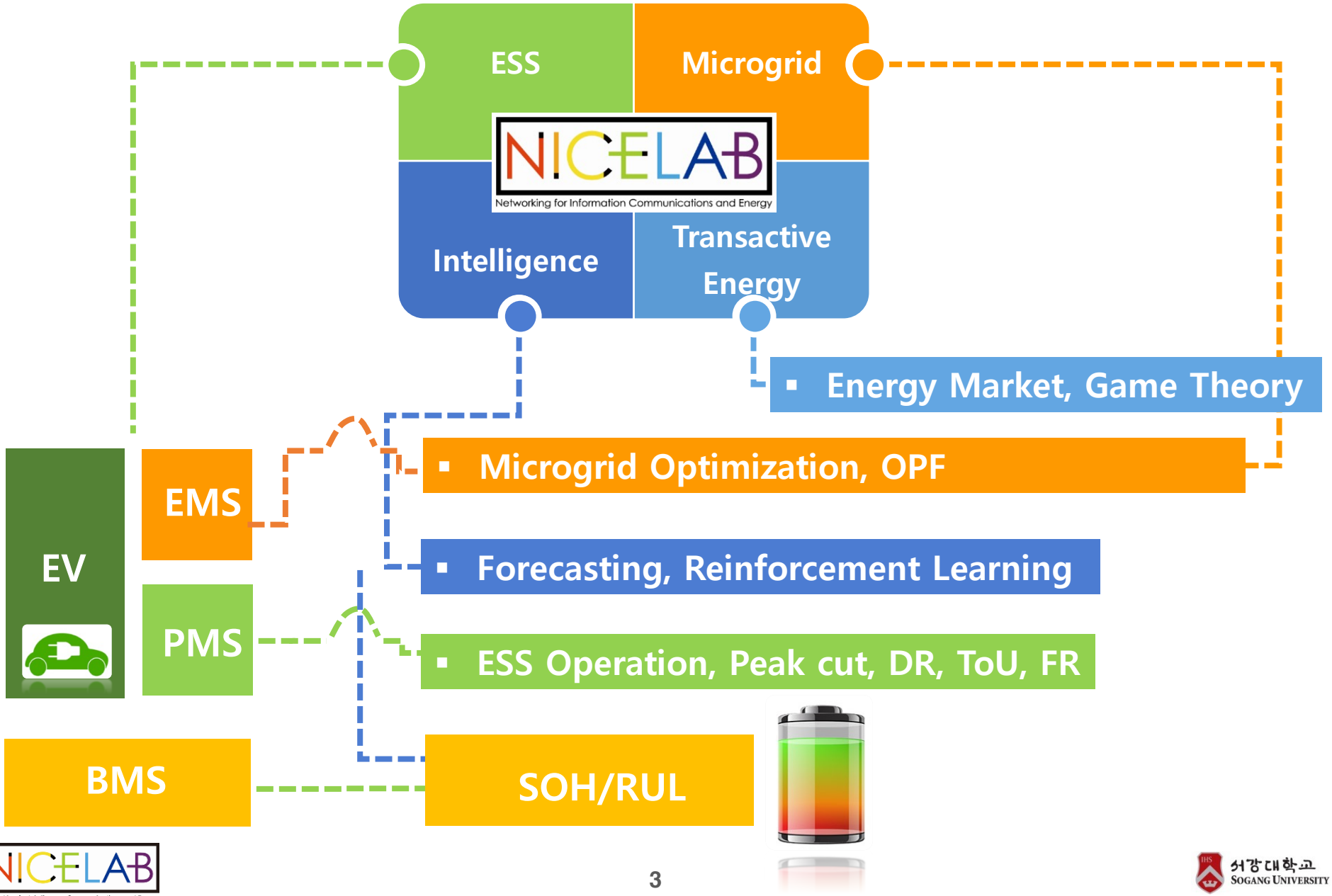


What starts at NICELAB changes the world

Networking for Intelligence Communications and Energy

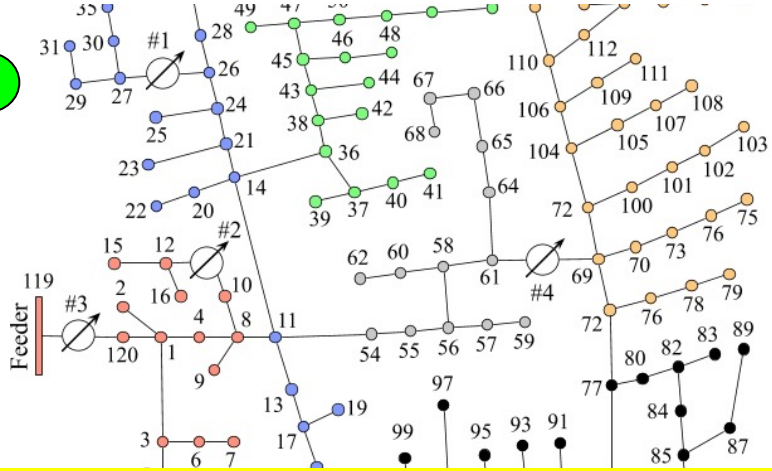


Research ITEM of NICE LAB

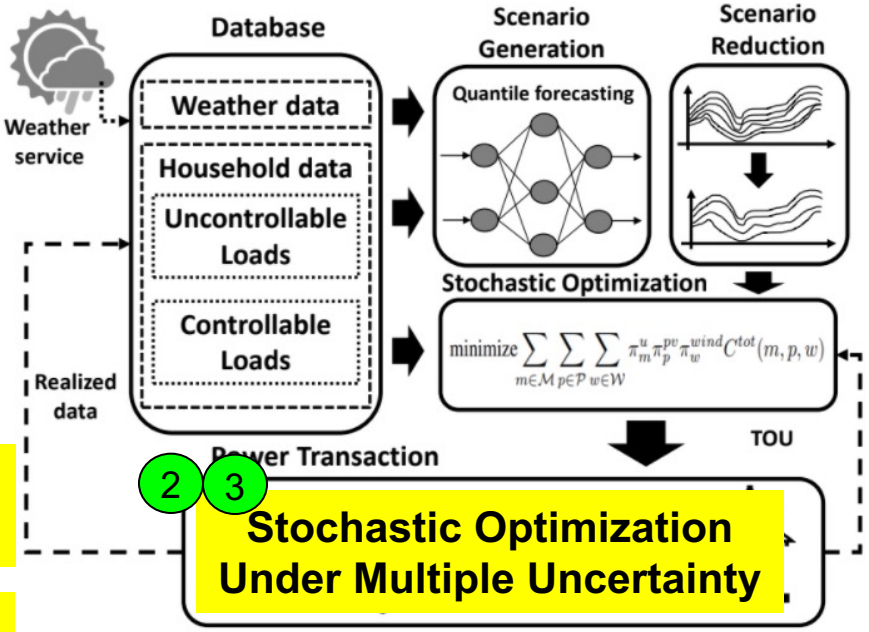


Thrust 1: Grid Optimization

1



Optimal Power Flow
Deep Learning



2 3 **Stochastic Optimization Under Multiple Uncertainty**

1 **Distributed Grid Optimization Using Alternating Direction Method of Multiplier (ADMM)**

$$\begin{array}{ll}\text{minimize} & f(x)+g(z)\\ \text{subject to} & Ax+Bz=c\end{array}$$

$$L_\rho(x,z,y)=f(x)+g(z)+y^T(Ax+Bz-c)+(\rho/2)\|Ax+Bz-c\|_2^2.$$

ADMM consists of the iterations

$$x^{k+1}:=\operatorname*{argmin}_x L_\rho(x,z^k,y^k) \quad (3.2)$$

$$z^{k+1}:=\operatorname*{argmin}_z L_\rho(x^{k+1},z,y^k) \quad (3.3)$$

$$y^{k+1}:=y^k+\rho(Ax^{k+1}+Bz^{k+1}-c), \quad (3.4)$$

Click!

1 **MG-SPICE, 1000x speed acceleration to solve OPF using ADMM**

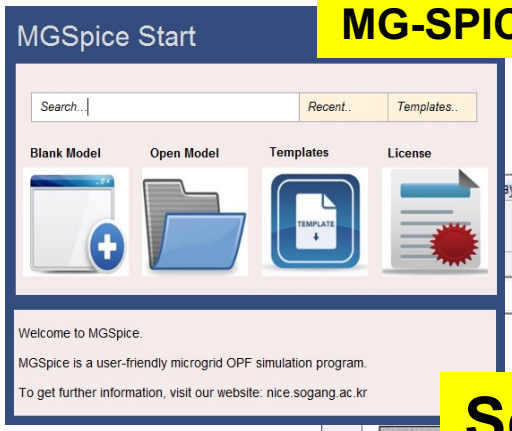
MG-SPICE: World-First MG Optimization Design Tool

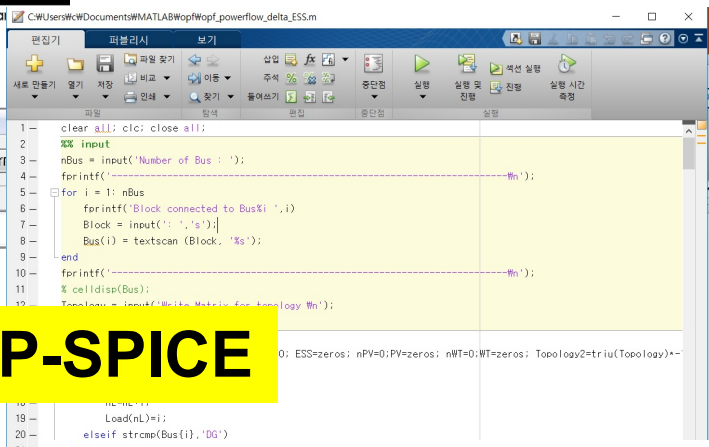
Micro Grid Simulation Program with Integrated Control Emphasis

MG-SPICE developed by NICE LAB

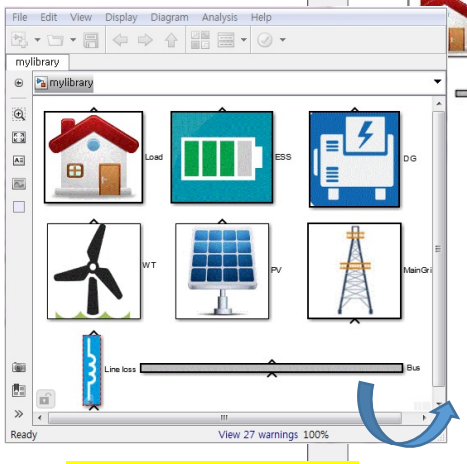
Networking for Information Communications at

Auto-Code Generation

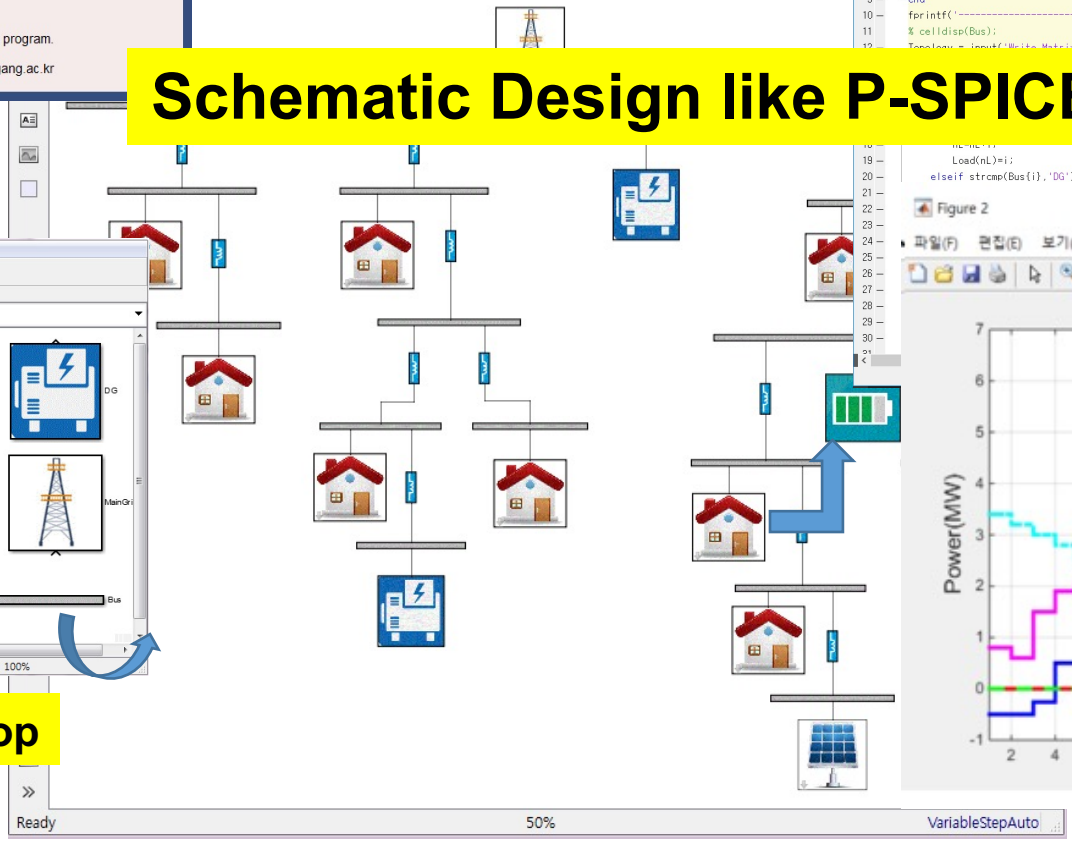


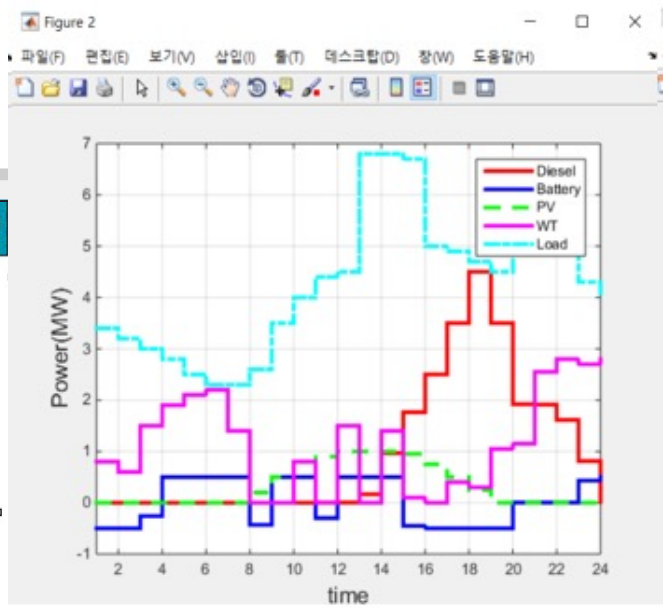


Schematic Design like P-SPICE

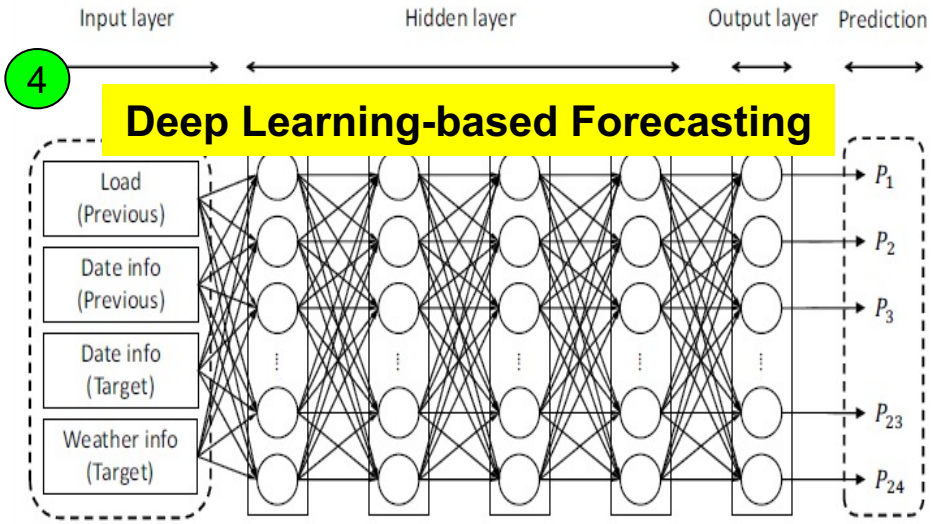


Drag & Drop

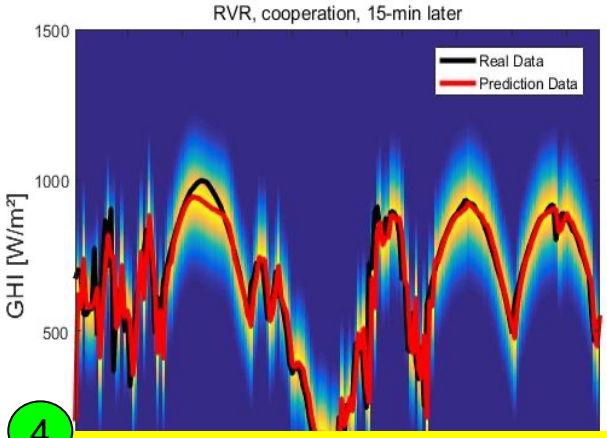




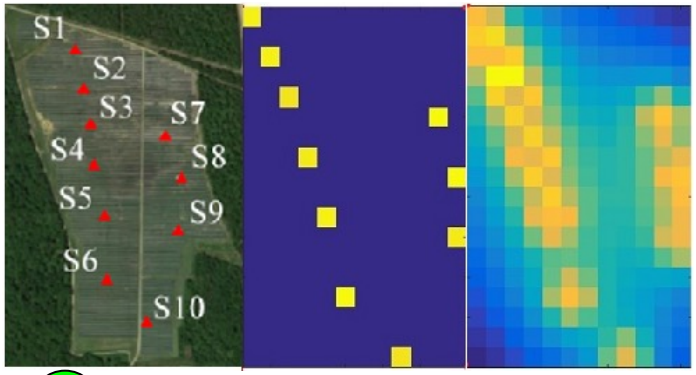
Thrust 2: Energy AI



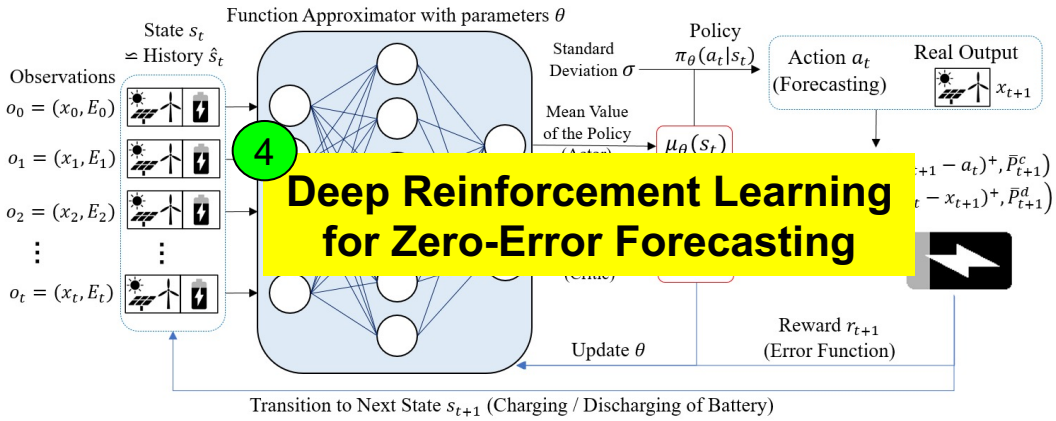
4 **Deep Learning-based Forecasting**



4 **Probabilistic PV Prediction Using Machine Learning**

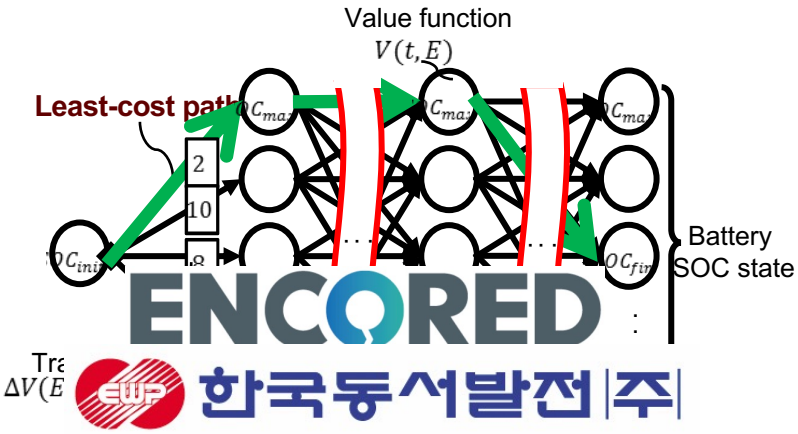


4 **CNN-based Cooperative Multi-Site PV Prediction**

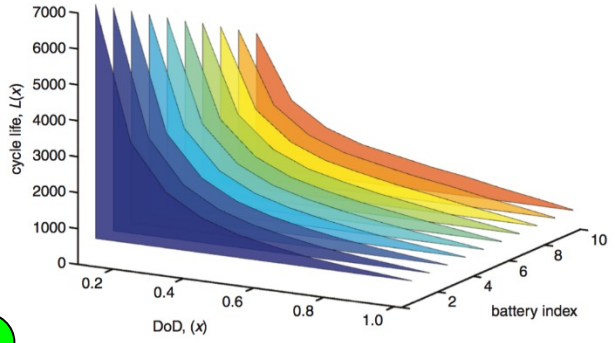


4 **Deep Reinforcement Learning for Zero-Error Forecasting**

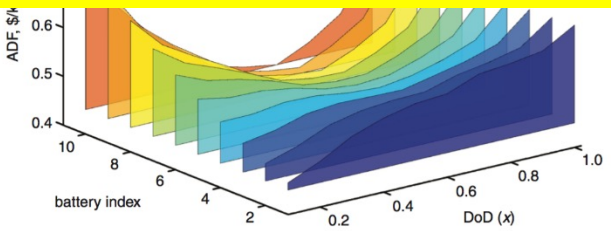
Thrust 3: EV/ESS Optimization/AI

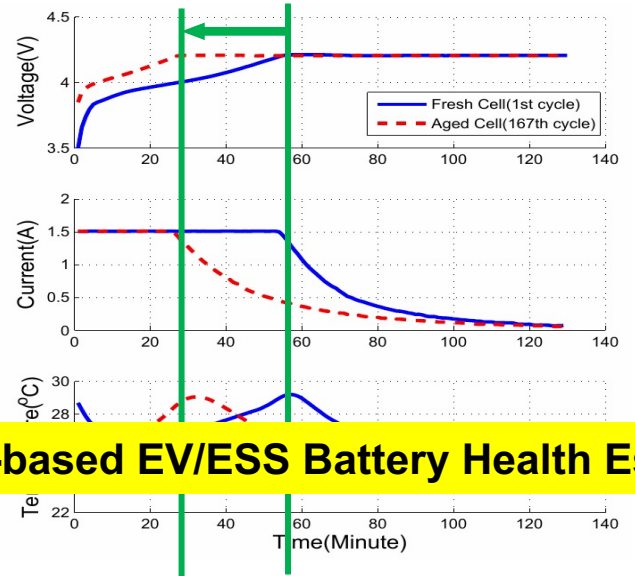


6 **Multi-Purpose ESS: DR, Peak-cut, ToU, Battery Degradation**

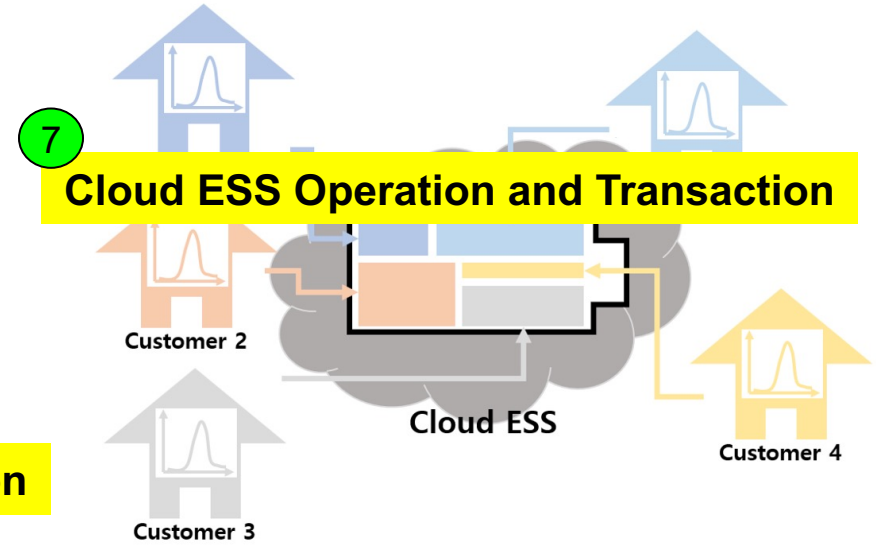


6 **Battery Degradation Modeling for EV/ESS**



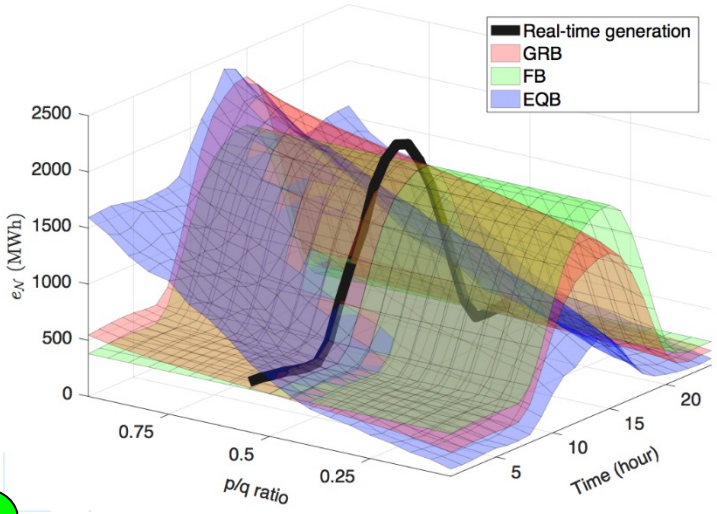


7 **AI-based EV/ESS Battery Health Estimation**



7 **Cloud ESS Operation and Transaction**

Thrust 4: Economics and Game Theory

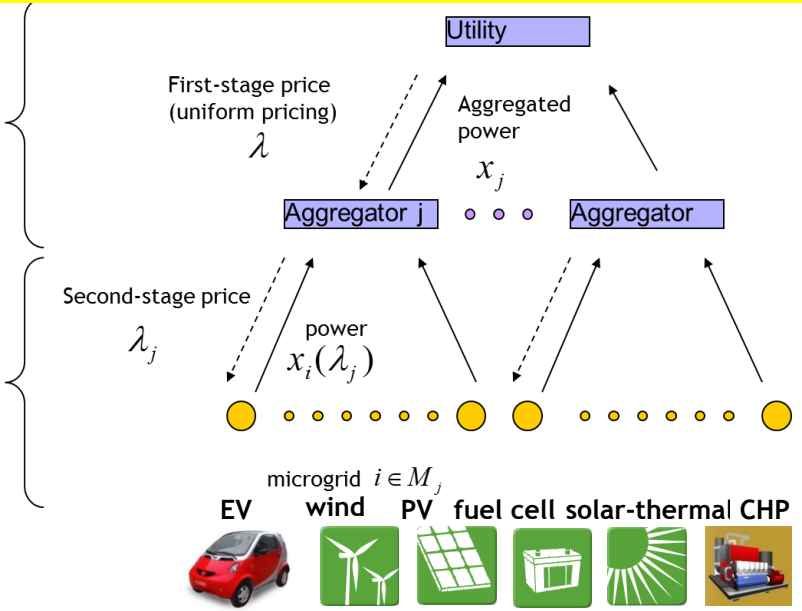


8

Cooperative Bidding Strategy in Renewable Energy Market

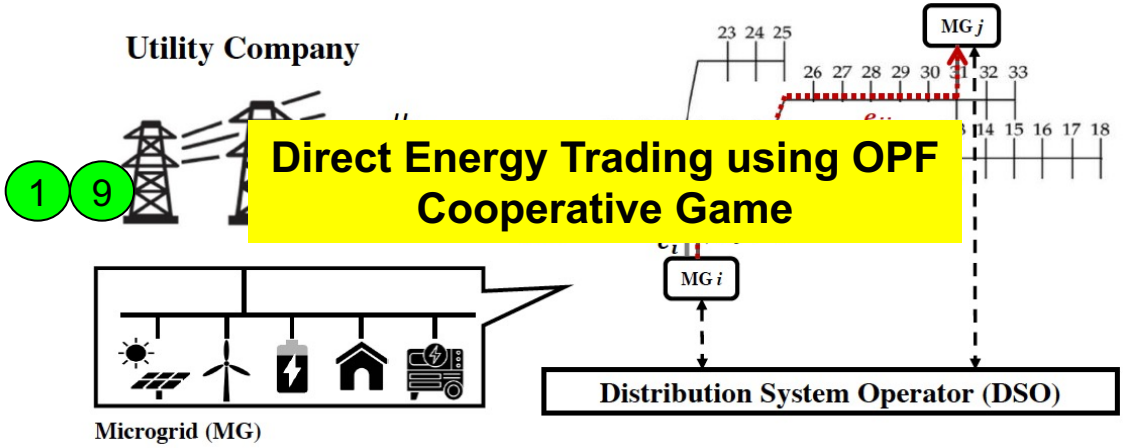
18

Market Model Using Stackelberg Game



19

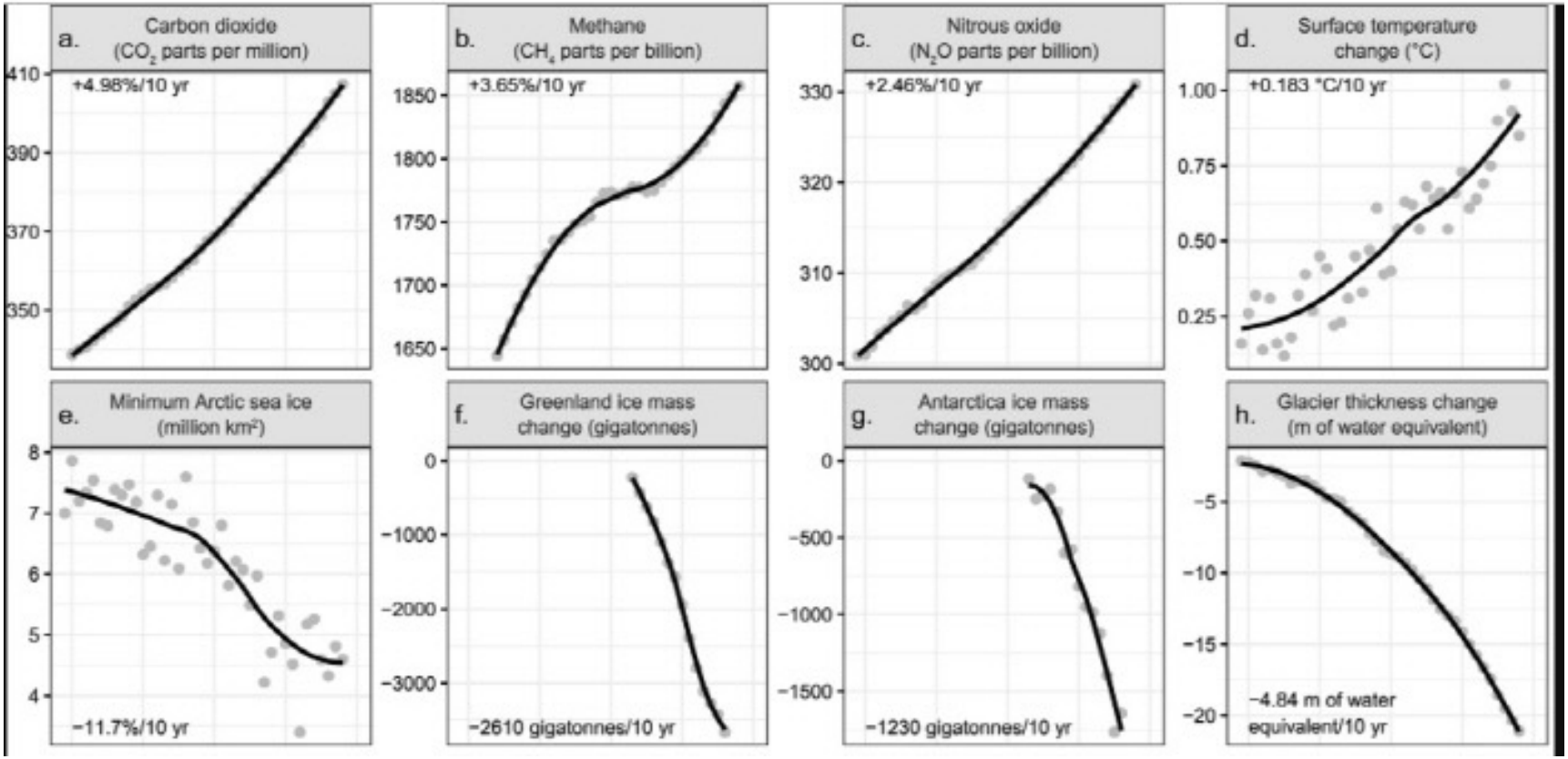
Direct Energy Trading using OPF Cooperative Game



Upcoming Challenges

Climate Change for past 40 years

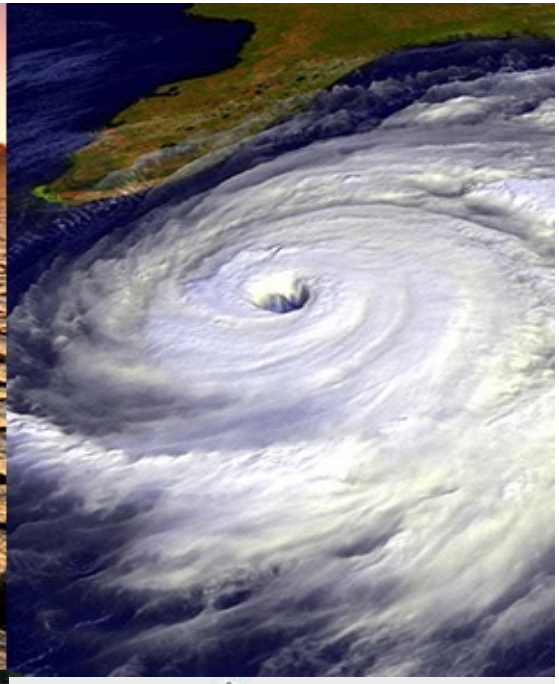
Carbon dioxide **Methane** **Nitrous oxide** **Surface temperature**



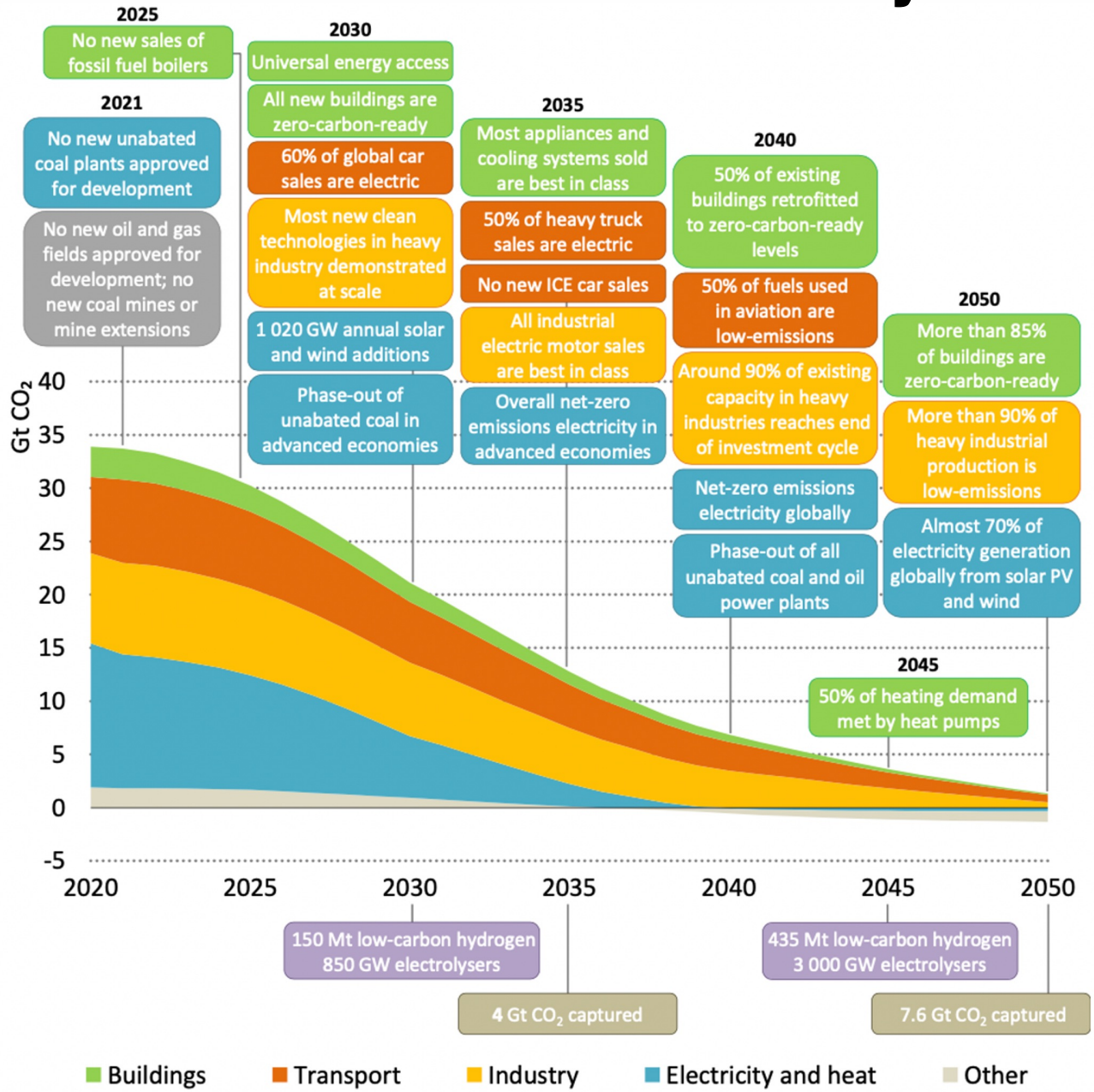
Arctic sea ice **Greenland ice** **Antarctica ice** **Glacier thickness**

- http://news.khan.co.kr/kh_news/khan_art_view.html?artid=201911061521001&code=970100

Climate Change

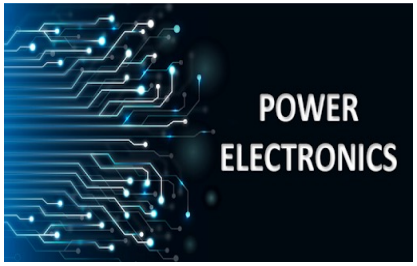


How to achieve Net-Zero by 2050?



https://www.weforum.org/agenda/2021/05/net-zero-emissions-2050-iea/

Renewable E & Electrification



NICELAB
Networking for Intelligence Communications and Energy

Toward Net-Zero 2050

☐ Renewable Generation

- Tremendous amount (500GW?) of PV/wind by 2050
- Fluctuation of renewable generation and demand

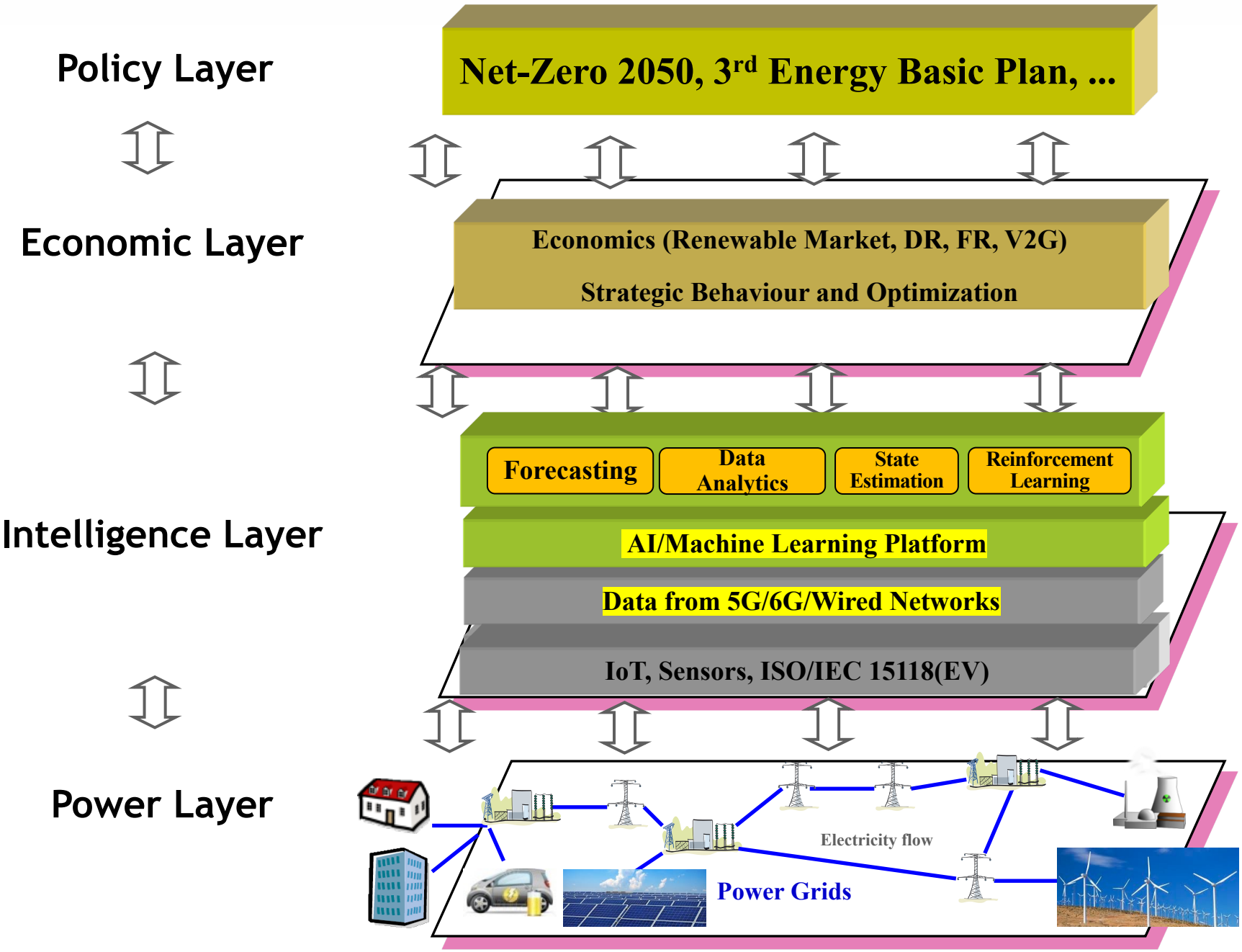
☐ Energy Forecasting

- Renewable generation forecasting
- Short-term load forecasting
- Energy data analytics

☐ Control

- Flexibility
- Optimal power flow
- Direct Energy Trading (P2P)

Layering Architecture for Future Grid



DeepComp: Deep Reinforcement Learning based Renewable Energy Error Compensable Forecasting

서강대학교
SOGANG UNIVERSITY

NICELAB
Networking for Intelligence Communications and Energy

Introduction

☐ Uncertainty of Renewable Power Generation

- Renewable power outputs heavily depend on weather conditions

☐ Conditions for the Power Grid Stability

- Power Generation = Power Demand
  - » Mismatch incurred by either over-generation or under-generation is harmful

☐ Importance of Renewable Energy Forecasting

- The proper amount of electricity supply should be prepared in advance
  - » Many power plants take some time to get to their desired output
  - » Increasing reserve for real-time control is a costly option
- A few minutes to one hour ahead forecasting is essential for adjusting imbalances through re-dispatch

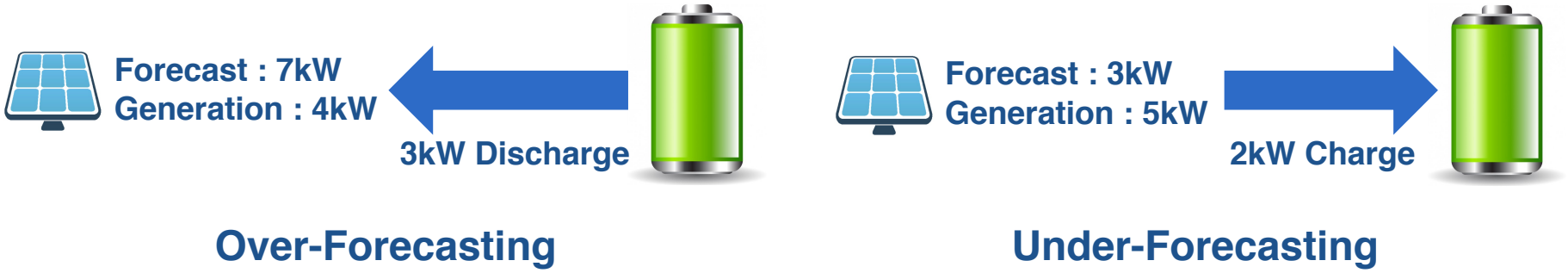
Introduction

☐ Deep Learning for Renewable Energy Forecasting

- Deep neural networks (DNNs) such as long short-term memory (LSTM) are widely used to improve forecasting accuracy

☐ Error Compensation using Batteries

- Forecasting always induces errors, and large-scale batteries can be used to compensate forecasting errors
  - » They can provide fast response to compensate forecasting errors



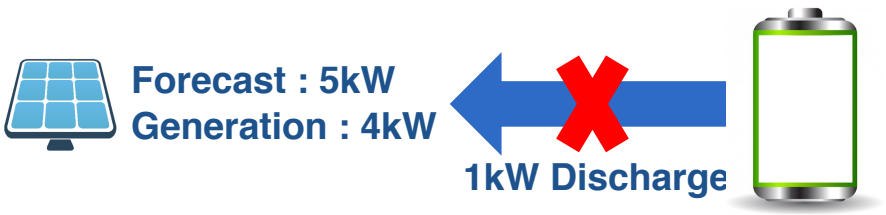
Over-Forecasting

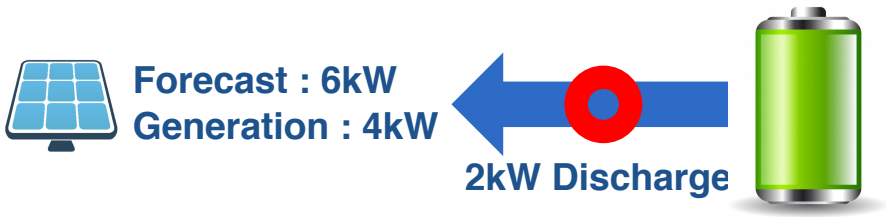
Under-Forecasting

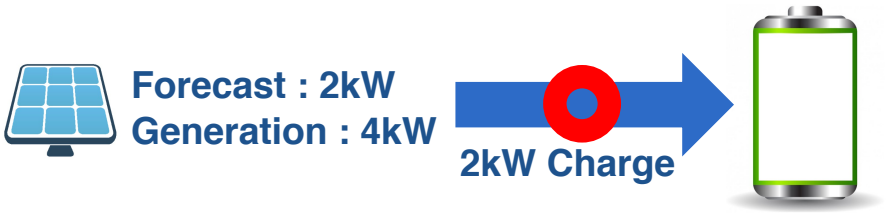
Introduction

☐ Objective of Forecasting Algorithms

- **Traditional deep learning-based forecasting methods commonly aim to minimize the forecasting errors**
  - » **Mean absolute error (MAE) or mean squared error (MSE) are used**
- **However, reducing errors does not necessarily imply compensable errors**







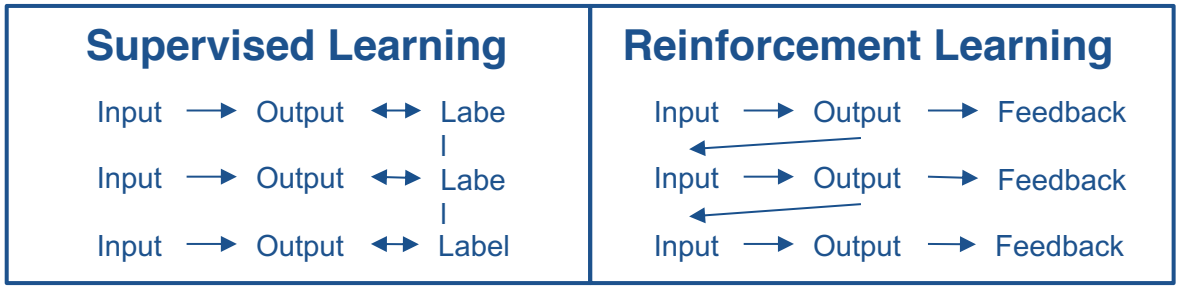
Introduction

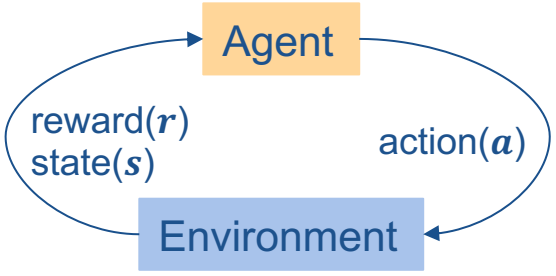
☐ Error Compensable Forecasting (DeepComp)

- We switch the objective of forecasting from reducing errors to making errors compensable

☐ Deep Reinforcement Learning (DRL)

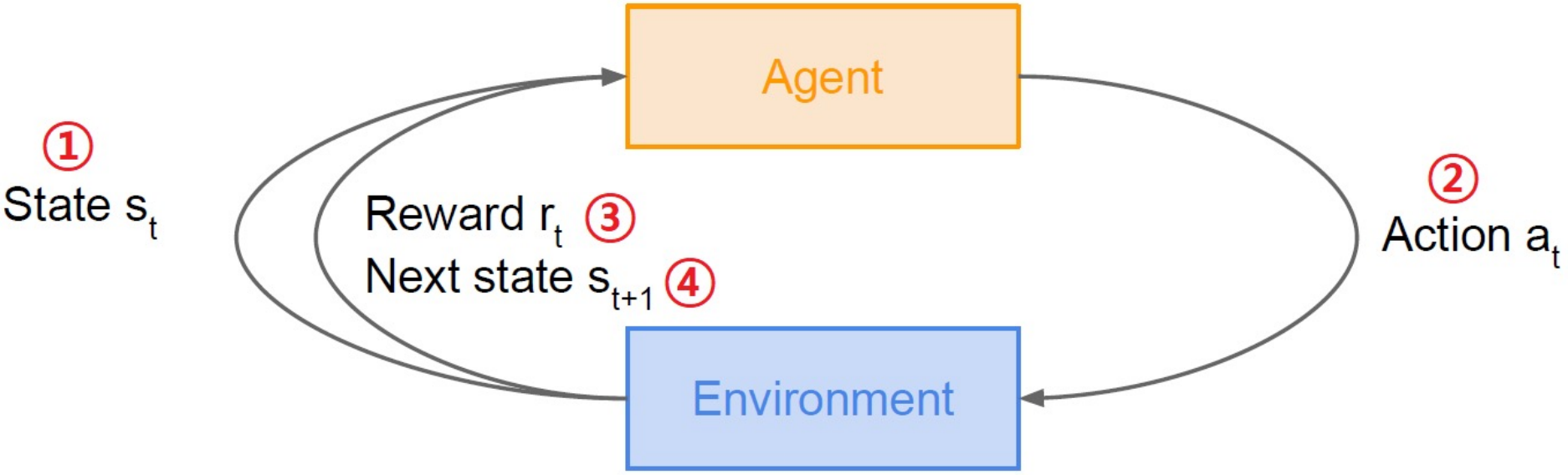
- The challenging part is the stored energy at current time affected by the previous forecasting results
- We tackle this problem by leveraging DRL





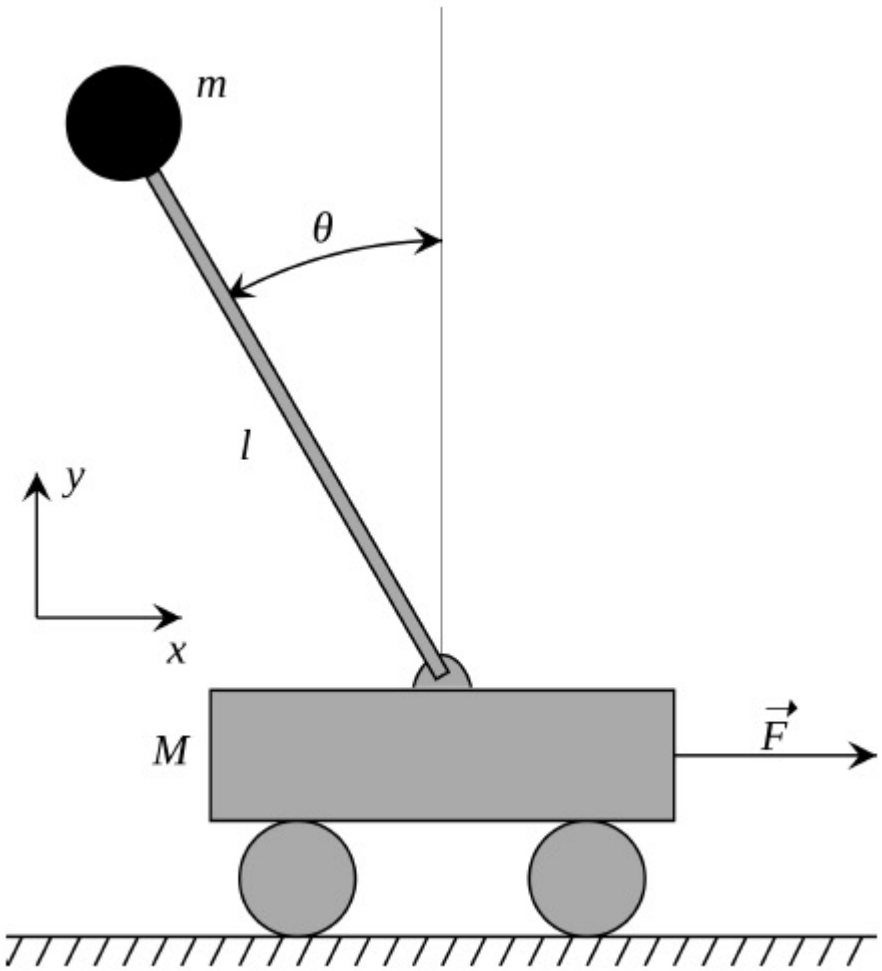
RL has interaction between an agent and the environment

Reinforcement Learning



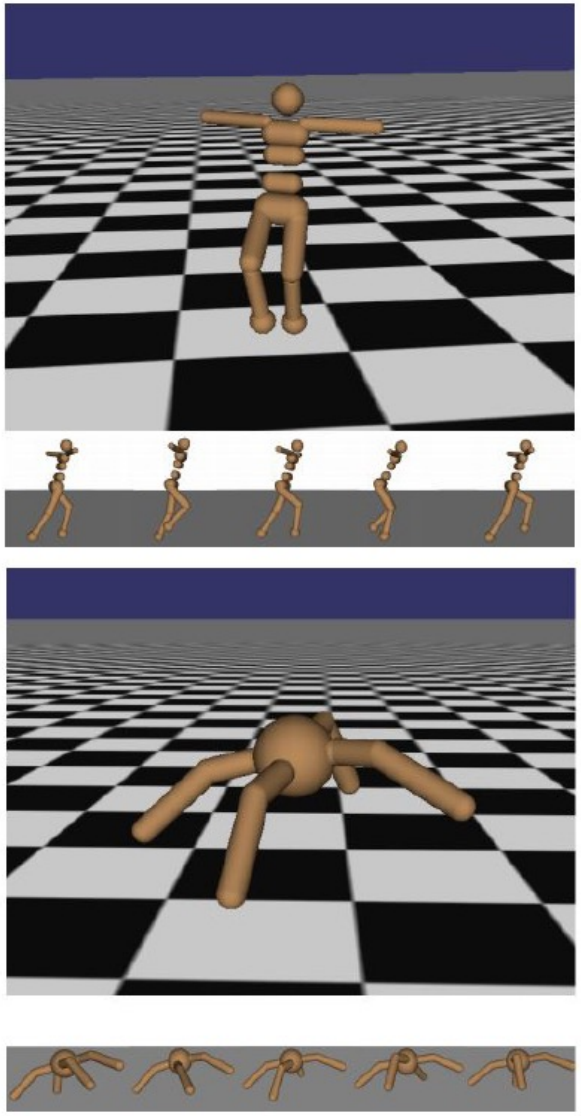
인생의 거의 모든 문제는 Sequential Decision Making under Uncertainty

Cart-Pole Problem



- **Objective**
  - Balance a pole on top of a movable cart
- **State**
  - Angle, angular speed, position, horizontal velocity
- **Action**
  - Horizontal force applied on the cart
- **Reward**
  - 1 at each time step if the pole is upright

Robot Locomotion

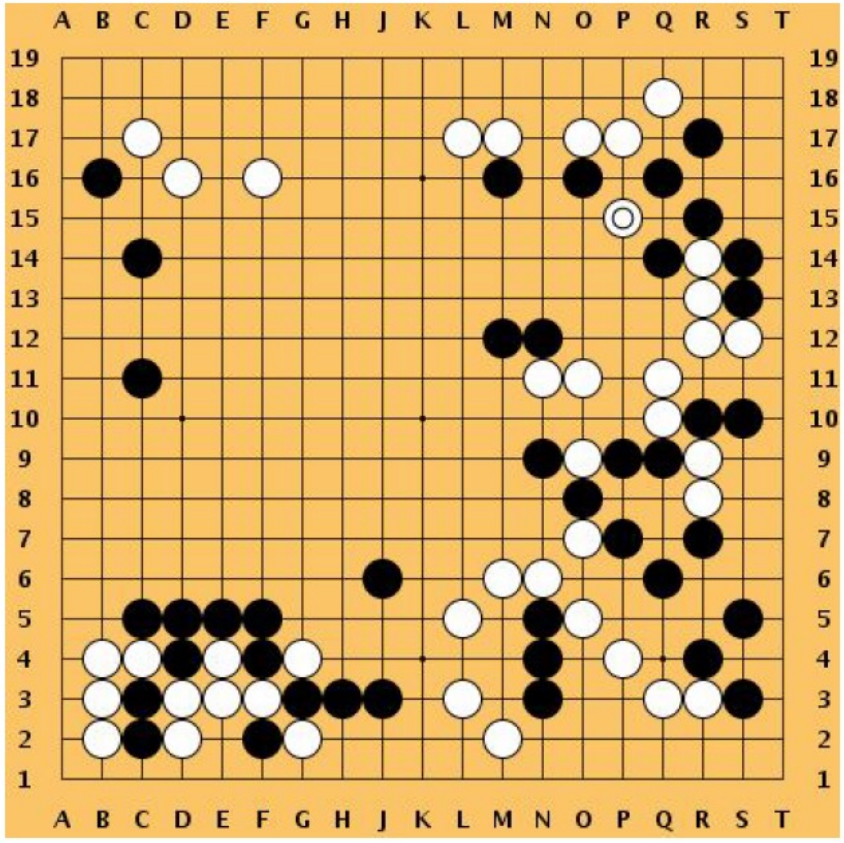


- **Objective**
  - Make the robot move forward
- **State**
  - Angle and position of the joints
- **Action**
  - Torques applied on joints
- **Reward**
  - 1 at each time step upright + forward movement

Atari Games

- **Objective**
  - Complete the game with the highest score
- **State**
  - Raw pixel inputs of the game state
- **Action**
  - Game controls e.g. Left, Right, Up, Down
- **Reward**
  - Score increase/decrease at each time step

Go



- **Objective**
  - Win the game
- **State**
  - Position of all pieces
- **Action**
  - Where to put the next piece down
- **Reward**
  - 1 if win at the end of the game, 0 otherwise

AlphaMoKu

☐ Made in NICE LAB using AlphaGo-Zero algorithm

```
76 if __name__ == '__main__':
77     run()
```



```
AI is preparing for a competition...
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1 class Human(object):
2     """
3     human player
4     """
5
6     def __init__(self):
7         self.player = None
```

AI win!

AlphaMogu

□ 2라운드

```
76 if __name__ == '__main__':
77     run()
```



```
AI is preparing for a competition...
```

```
]: 1
```

```
]: 1
```

```
]: 1
```

```
]: 1
```

```
]: 1
```

```
]: 1
```

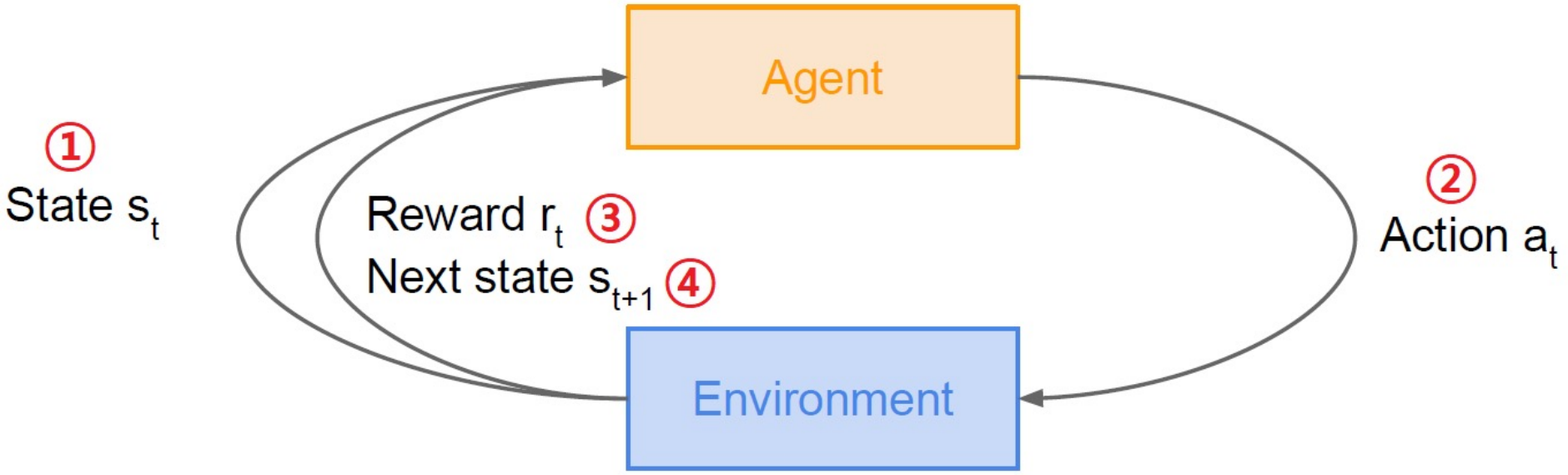
```
]: 1
```

```
]: 1
```

```
]: 1
```

```
]:
1  class Human(object):
2      """
3      human player
4      """
5
6      def __init__(self):
7          self.player = None
8
9      def set_player_ind(self, p):
```

How can we mathematically formalize the RL problem?



MDP는 RL에 관해 수학적으로 아름다운 솔루션을 줄 수 있음

Markov Decision Process

- Mathematical formulation of the RL problem
- Markov property
  - Current state completely characterizes the state of the world
  - Defined by $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$
    - $\mathcal{S}$: set of states
    - $\mathcal{A}$: set of actions
    - $\mathcal{R}$: distribution of reward given (state, action) pair
    - $\mathbb{P}$: state transition probability given (state, action) pair
    - γ: discount factor

Markov Decision Process

- At time step $t = 0$, environment samples initial state $s_0 \sim p(s_0)$
- Then, at each time step t
  - Agent selects action a_t
  - Environment samples reward $r_t \sim \mathcal{R}(\cdot \mid s_t, a_t)$
  - Environment samples next state $s_{t+1} \sim \mathbb{P}(\cdot \mid s_t, a_t)$
  - Agent receives reward r_t and goes to next state s_{t+1}

- A policy π is a function from $\mathcal{S}$ to $\mathcal{A}$ that specifies what action to take in each state

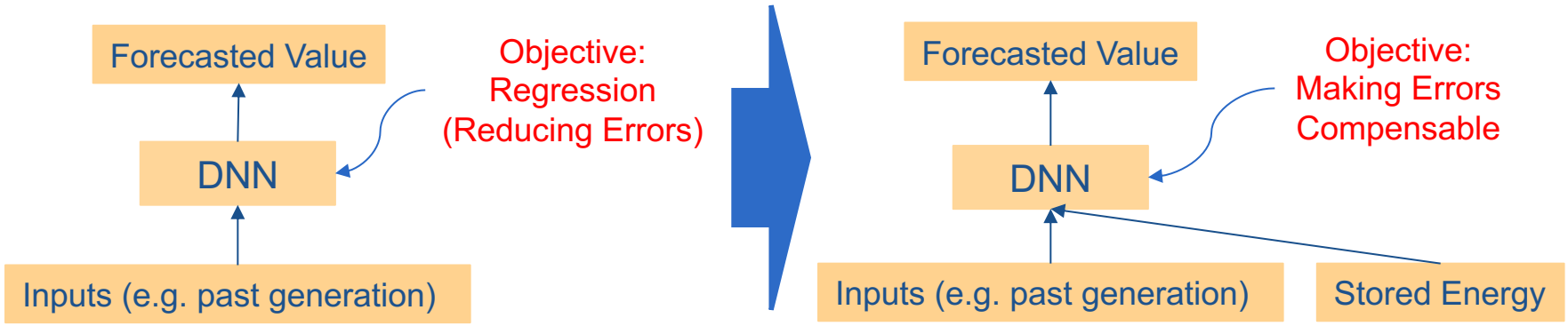
- Objective
  - Find policy π^* that maximizes cumulative discounted reward

$$\sum_{t \geq 0} \gamma^t r_t$$

Introduction

☐ Proposed DeepComp Framework

- **Use DNN architectures of traditional forecasting**
  - » Inputs of DNN for forecasting + Currently stored energy → State
  - » Outputs of DNN for forecasting (Forecasted value) → Action
- **Proximal Policy Optimization (PPO)**
  - » An action is a continuous forecasted value
  - » To enable continuous control, we leverage PPO that is simple to implement with outstanding performance



Traditional Forecasting **Proposed Forecasting**

System Model

☐ Battery Operation

- **Constraints of the stored energy at time slot t (E_t)**

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

- **Charging and discharging power limitation ($\bar{P}^c_{t+1}$ and $\bar{P}^d_{t+1}$) in the next time slot can be obtained as follows**

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

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

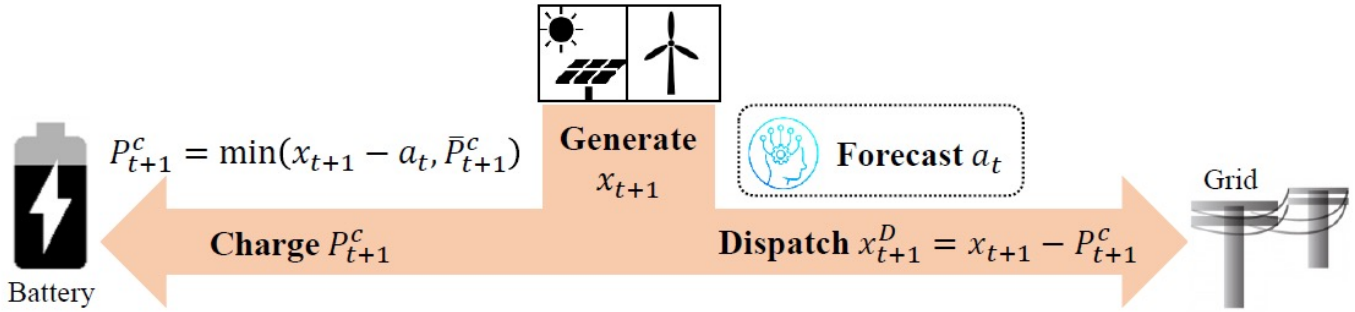
$E_{\max}$	Maximum Battery Capacity
$\mathrm{SoC}_{\min}$	Minimum State-of Charge
$\mathrm{SoC}_{\max}$	Maximum State-of Charge

Δt	Time Duration
$P^c_{\max}$	Maximum Charging Power
$P^d_{\max}$	Maximum Discharging Power
η_c	Charging Efficiency
η_d	Discharging Efficiency

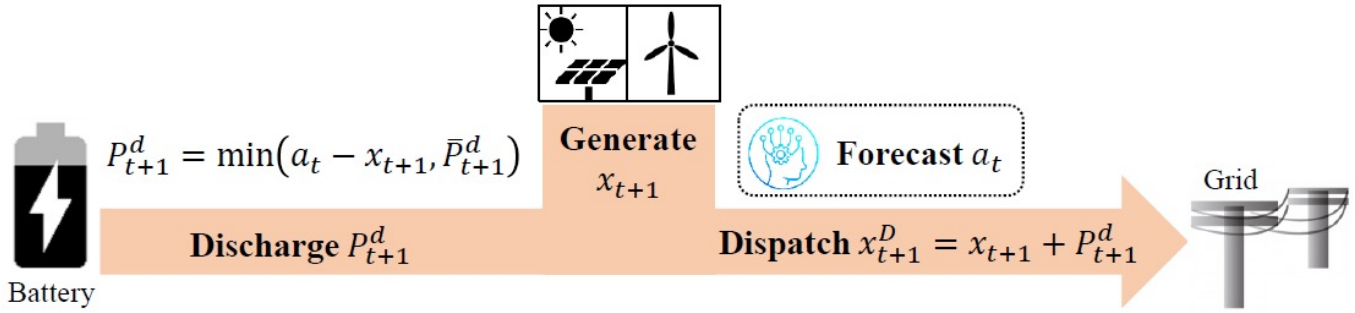
System Model

☐ Battery Operation

- x_t: Real generation value in time slot t
- a_t: Forecasted value in the next time slot $t+1$



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



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

System Model

☐ Error Function

- **Dispatched Error**
  - » The critical aspect of power grid operation is the error between forecasting and dispatched values

$$e_{t+1}^{D} = a_t - x_{t+1}^{D}$$

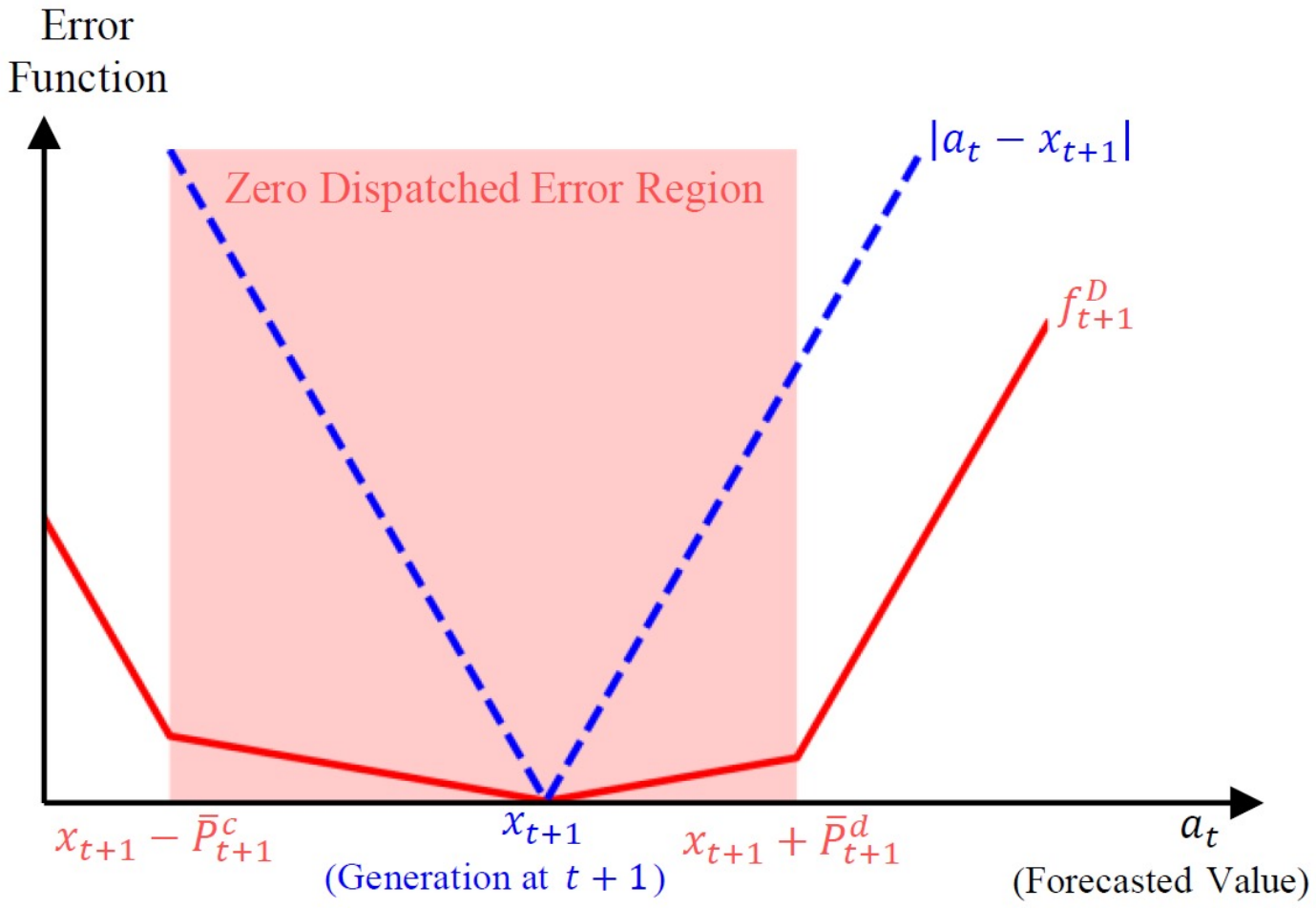
- **Error Function**
  - » Our primary objective is to minimize $\left|e_{t+1}^{D}\right|$
  - » If $e_{t+1}^{D} = 0$ is possible, we aim to minimize the use of battery to avoid battery degradation as well as energy transfer loss

$$f_{t+1}^{D} = \left|e_{t+1}^{D}\right| + \beta_c P_{t+1}^{c} + \beta_d P_{t+1}^{d}$$

System Model

☐ Error Function

$$f_{t+1}^{D} = \left| e_{t+1}^{D} \right| + \beta_c P_{t+1}^{c} + \beta_d P_{t+1}^{d}$$



NICELAB
Networking for Intelligence Communications and Energy

System Model

☐ Main Problem

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

- E_{init}: Initially stored energy
- $\gamma \in (0, 1)$: Discount factor (Determines the importance of future impacts)

☐ Sequential Decision Making under Uncertainty

- Machine learning techniques are required as we do not know x_{t+1} at t
- There exists time-coupling because of the stored energy

$$E_{t+1} = E_t + \eta_c P_{t+1}^{c} \Delta t - \frac{1}{\eta_d} P_{t+1}^{d} \Delta t$$

Proposed Methodology

☐ Markov Decision Process (MDP)

- Input: state(s_t)
- Output: action(a_t)
- Feedback: reward(r_{t+1}), next state(s_{t+1})
- A state contains all useful information from the history
  - » $\mathbb{P}(s_{t+1}, r_{t+1}|s_t, a_t) = \mathbb{P}(s_{t+1}, r_{t+1}|s_0, a_0, \cdots, s_t, a_t)$
- The action a_t is generated by the policy π
  - » $\pi(a_t|s_t)$: probability (density) of taking action a_t at state s_t
- Value is the total expected discounted reward.
  - » $V^{\pi}(s) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots | s_t = s, \pi\right]$
  - » $Q^{\pi}(s, a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots | s_t = s, a_t = a, \pi\right]$
- Our objective is to find a policy that maximize value

Proposed Methodology

☐ Main Problem

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

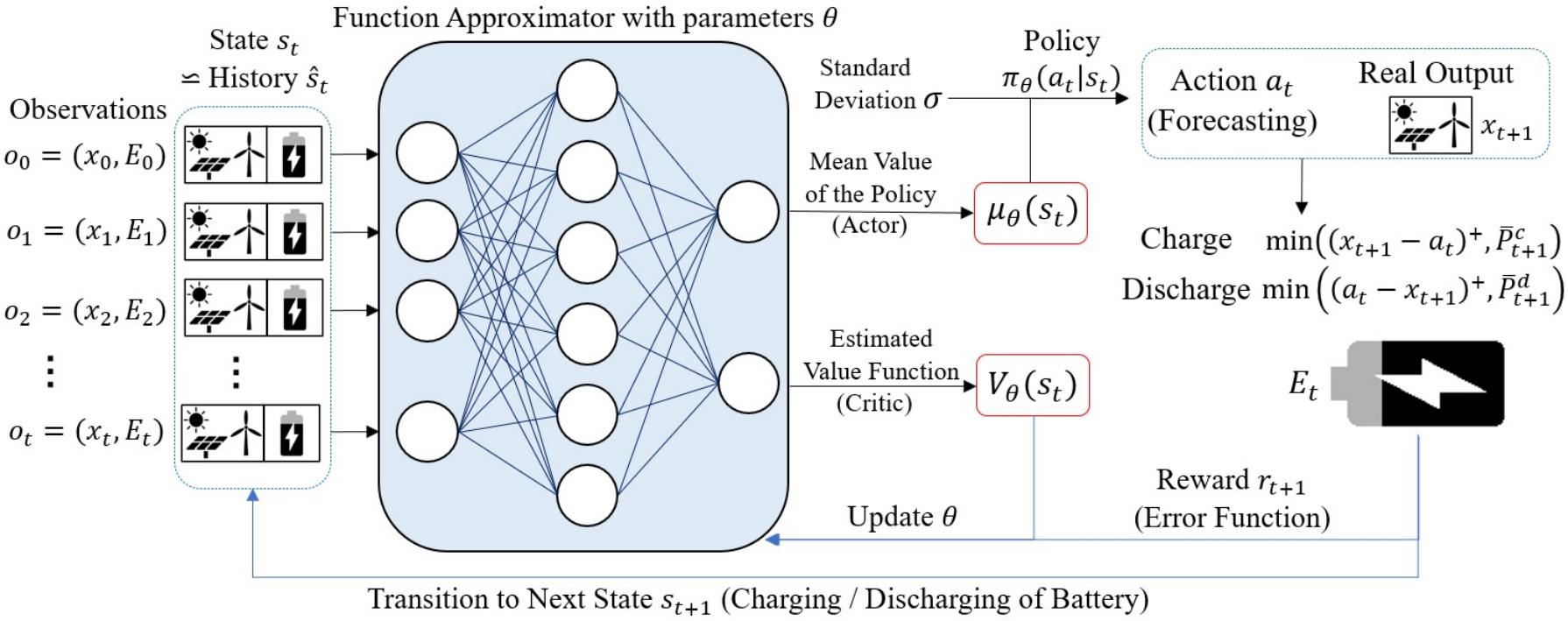
- » Reward $r_{t+1} = -f_{t+1}^{D}$ (Negation of the Error Function)

- r_{t+1} is determined by x_{t+1}, E_t, and a_t
  - » State $s_t = (x_{t+1},\ E_t)$
  - » Action a_t: Forecasted value in the next time slot $t+1$
- Partially Observable: x_{t+1} is unknown at time slot t
  - » Observation $o_t = (x_t,\ E_t)$
- The entire history of observation and action pairs is required
- a_{t-1} is determined by E_{t-1}, x_t, and E_t (o_{t-1} and o_t)
  - » State $(o_0, o_1, \cdots o_{t-1}, o_t)$

Proposed Methodology

☐ Policy-based RL

- **The policy $\pi_\theta(a_t|s_t)$ is directly parameterized with parameters θ**
  - » **In general, $\pi_\theta(a_t|s_t)$ is captured by Gaussian distribution**
  - » **The function approximator outputs its mean $\mu_\theta(s_t)$**
  - » **Standard deviation σ is fixed and tuned during the training**



Proposed Methodology

☐ Error Compensable Forecasting Problem with Policy-based RL

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

- $\sum_{t=0}^{\infty} \gamma^t r_{t+1}$ corresponds to the value function $V^{\pi_\theta}(s_0)$
- $J_\pi(\theta)$ is the expected value function (objective function)

☐ Policy Gradient Theorem

$$\nabla_\theta J_\pi(\theta) = \mathbb{E}_{s\sim\rho_\theta, a\sim\pi_\theta} \left[\frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} Q^{\pi_\theta}(s,a) \right]$$

» ρ_0: Initial state distribution ($\rho_0(s) = p(s_0 = s)$)

» ρ_θ: Discounted state distribution by following π_θ

$$\rho_\theta(s) = p(s_0 = s) + \gamma p(s_1 = s) + \gamma^2 p(s_2 = s) + \gamma^3 p(s_3 = s) + \cdots$$

Proposed Methodology

☐ DeepComp Problem with Trust Region Constraints

$$\max_{\theta} \quad \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}},a\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}Q^{\pi_{\theta_{\text{old}}}}(s,a)\right]$$

$$\text{s.t.} \quad \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\;||\;\pi_\theta(\cdot|s)\right)\right]\leq\varepsilon$$

☐ Trust Region Policy Optimization (TRPO)

- Constrain the updated policy θ to be quite similar with the previous policy θ_{old} using KL divergence
- It prevents overfitting to the most recently sampled data by preventing excessive update
- TRPO is relatively complicated to solve because of the constraint

Proposed Methodology

☐ Proximal Policy Optimization (PPO)

- It solves TRPO easily by clipping the ratio at $1+\epsilon$ or $1-\epsilon$
- The agent empirically runs the policy π_θ to sample data
  - » Sample for T time steps: $(s_0, a_0, r_0, s_1, \cdots, r_{T-1}, s_T)$
- Clipped Surrogate Objective

$$\hat{J}_\pi^{\text{clip}}(\theta) \triangleq \hat{\mathbb{E}}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t,\ g(\epsilon, \hat{A}_t)\right)\right] \qquad g(\epsilon, \hat{A}_t) = \begin{cases}(1+\epsilon)\hat{A}_t & \text{if } \hat{A}_t \geq 0,\\ (1-\epsilon)\hat{A}_t & \text{if } \hat{A}_t < 0,\end{cases}$$

  - » $\hat{\mathbb{E}}_t[\cdot]$ indicates the empirical average over $t \in [0, T-1]$
- Estimated Advantage Function $\hat{A}_t$
  - » Replace $Q^\pi(s,a)$ with advantage $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$
  - » This can reduce variance of policy gradient by making smaller gradients
  - » How to estimate?

Proposed Methodology

☐ Generalized Advantage Estimation (GAE)

- Estimate the advantage function $\widehat{A}_t$ with significantly reduced variance while maintaining a tolerable level of bias
- $\widehat{A}_t$ can be estimated using the sampled data $(s_0, a_0, r_1 \cdots r_T, s_T)$

$$\hat{A}_t = \sum_{k=0}^{T-t-1} (\gamma\lambda)^k \hat{\delta}_{t+k} \qquad \hat{\delta}_t \triangleq r_{t+1} + \gamma V_\theta(s_{t+1}) - V_\theta(s_t)$$

» $\lambda \in (0, 1)$: Parameter for the bias-variance tradeoff

- Critic calculates the estimated value function

$$\mathcal{L}_V(\theta) = \hat{\mathbb{E}}_t \left[\left(r_{t+1} + \gamma V_\theta(s_{t+1}) - V_\theta(s_t) \right)^2 \right]$$

☐ Main Objective

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

C: Coefficient of the critic

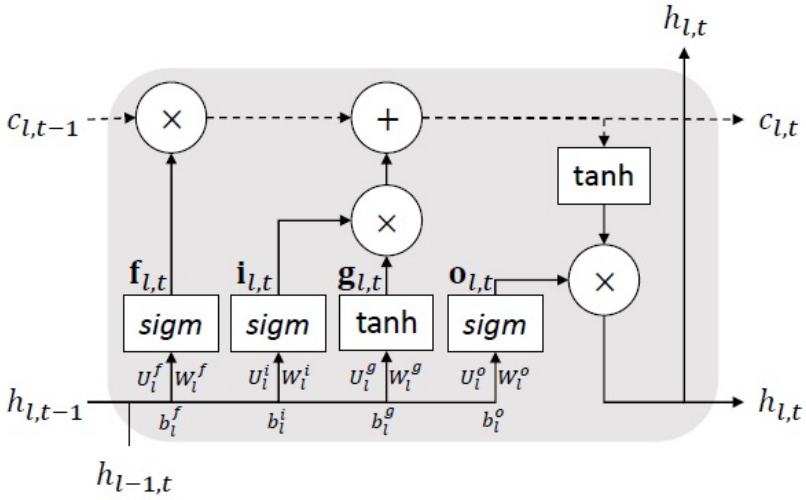
Proposed Methodology

☐ Deep Learning Architecture

- **LSTM can be used when the environment is partially observable**
  - » **LSTM is also widely used to forecast the renewable energy generation and show better performance than feedforward neural network (FNN)**
- **LSTM 1st layer input is made by the observation vectors $o_t = (x_t, E_t)$**

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

- **LSTM relation**



$$\mathbf{f}_{l,t} = sigm(U_l^f h_{l,t-1} + W_l^f h_{l-1,t} + b_l^f)$$
$$\mathbf{i}_{l,t} = sigm(U_l^i h_{l,t-1} + W_l^i h_{l-1,t} + b_l^i)$$
$$\mathbf{g}_{l,t} = \tanh(U_l^g h_{l,t-1} + W_l^g h_{l-1,t} + b_l^g)$$
$$\mathbf{o}_{l,t} = sigm(U_l^o h_{l,t-1} + W_l^o h_{l-1,t} + b_l^o)$$
$$c_{l,t} = \mathbf{f}_{l,t} \odot c_{l,t-1} + \mathbf{i}_{l,t} \odot \mathbf{g}_{l,t}$$
$$h_{l,t} = \mathbf{o}_{l,t} \odot \tanh\left(c_{l,t}\right)$$

Proposed Methodology

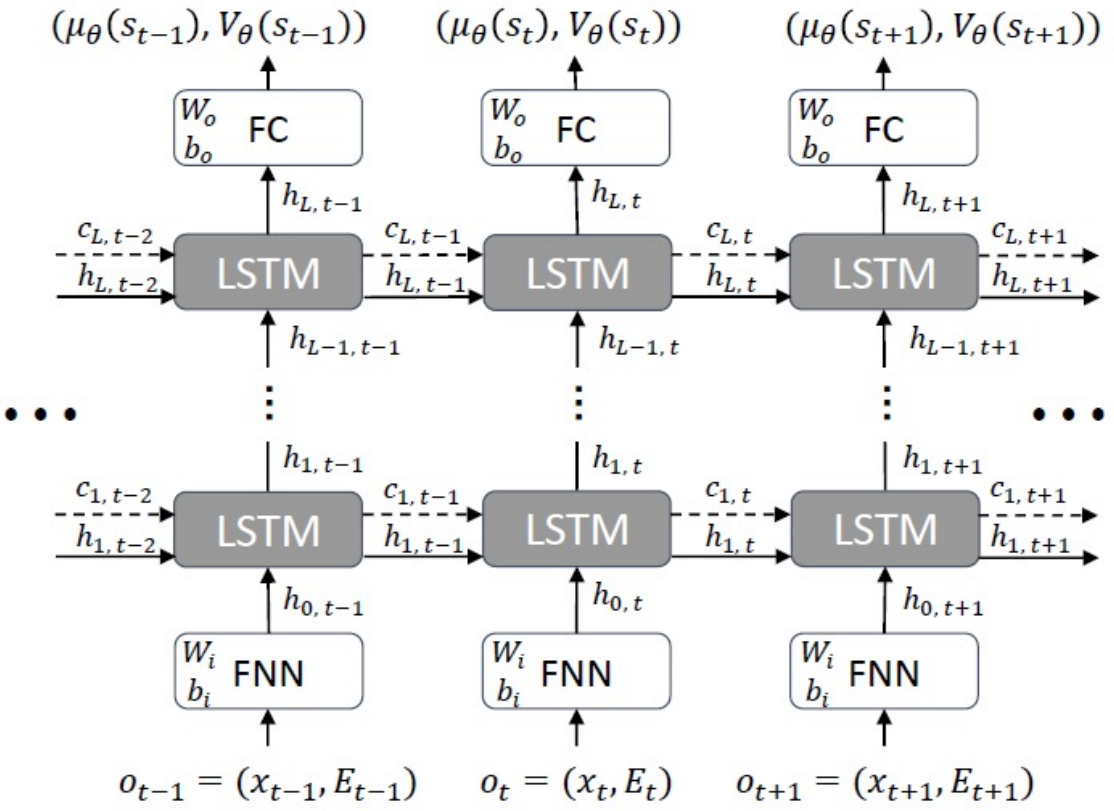
☐ LSTM Structure

- $c_{l,t-1}$ and $h_{l,t-1}$ contain the history of observations $(o_0, o_1, \cdots o_{t-1}, o_t)$
- **We obtain the state**

$$s_t = \left(\{c_{l,t-1}, h_{l,t-1}\}_{l=1}^{l=L}, o_t \right)$$

- **The output vector of the LSTM $h_{L,t}$ pass through the fully-connected (FC) layer**

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



Proposed Methodology

Algorithm

- At each iteration, the agent samples data $(s_0, a_0, r_0, s_1, \cdots, r_{T-1}, s_T)$ according to θ_{old}
- $\mathcal{L}(\theta)$ is minimized using the sampled data for K epochs with learning rate α
- θ_{old} is replaced with the updated θ, and this process is repeated until convergence

Algorithm 1: Proposed LSTM-based ECF 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
  $s_0 \leftarrow \left(\{c_{l,-1}, h_{l,-1}\}_{l=1}^{l=L}, o_0\right)$;
  for $t \leftarrow 0$ **to** $T-1$ **do**
    Output $(\mu_\theta(s_t), V_\theta(s_t))$ 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 $o_{t+1} \leftarrow (x_{t+1}, E_{t+1})$;
    $s_{t+1} \leftarrow \left(\{c_{l,t}, h_{l,t}\}_{l=1}^{l=L}, o_{t+1}\right)$;
  end
  $\theta_{\text{old}} \leftarrow \theta$;
  Output $V_\theta(s_T)$;
  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

Dataset

- 4-year (2016~2019) data provided by Belgium Elia
  - » Aggregated solar and wind data (Maximum generation: 100MW)
  - » The data are normalized between 0 and 1 using the installed PV generation capacity, and sampled every 1 hour
  - » We split the dataset into a training set (50%), a validation set (25%), and a test set (25%)

Battery Model

- We also normalize $E_{\max}$ (1 p.u. = 100MWh)
- Simulate with different $E_{\max}$ (0.1 p.u. ~ 0.5 p.u.)
- Parameters are summarized in the table

Battery related parameters	value
Δt	1 hour
η_c	0.9
η_d	0.9
$\mathrm{SoC}_{\max}$	0.9
$\mathrm{SoC}_{\min}$	0.1
$P^c_{\max}/E_{\max}$	1/3
$P^d_{\max}/E_{\max}$	1/3
β_c	0.01
β_d	0.01

Experimental Results

☐ Hyperparameter Selection

- Hyperparameters are determined based on the mean absolute percentage error (MAPE) of the dispatched error

$$\text{MAPE} = \frac{100}{|\mathcal{V}|} \sum_{t \in \mathcal{V}} \left| \frac{e_{t+1}^{D}}{x_{t+1}^{D}} \right| [\%]$$

$\mathcal{V}$: Validation dataset

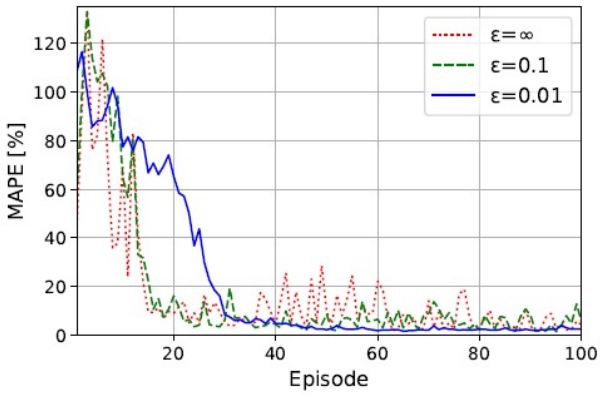
- All networks are trained based on the Adam optimizer with recommended learning rate $\alpha = 0.001$ and the minibatch size $T = 128$

☐ LSTM Model Selection

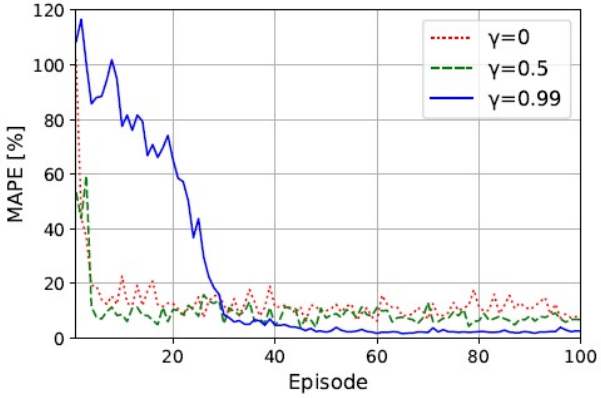
- We determine the model for ERF first, and use the same structure for the DeepComp
  - » Two hidden layers and 16 hidden neurons per each hidden layer

Experimental Results

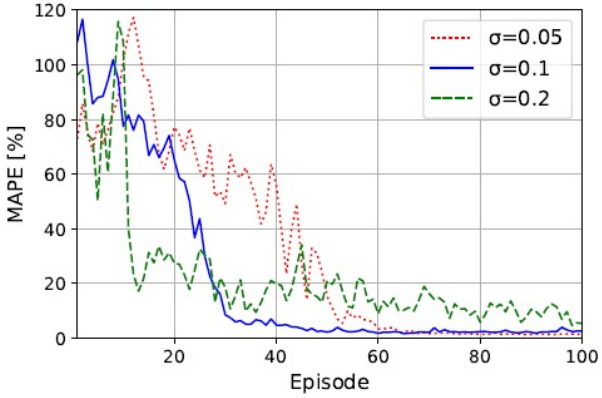
☐ PPO related hyperparameter Selection



(a) Hyperparameter for clipping (ϵ)

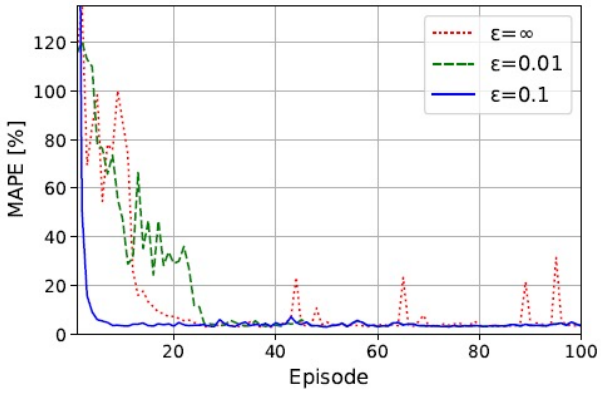


(b) Discount factor (γ)

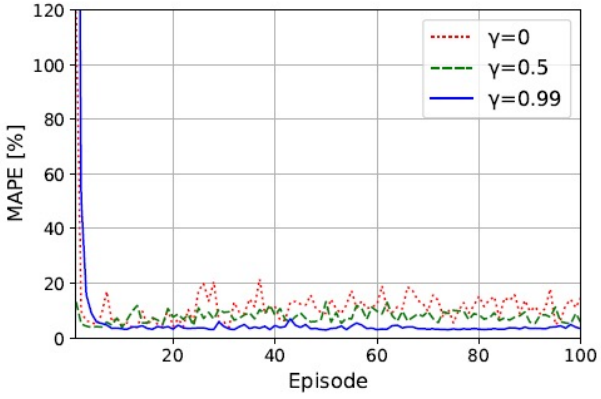


(c) Standard deviation (σ)

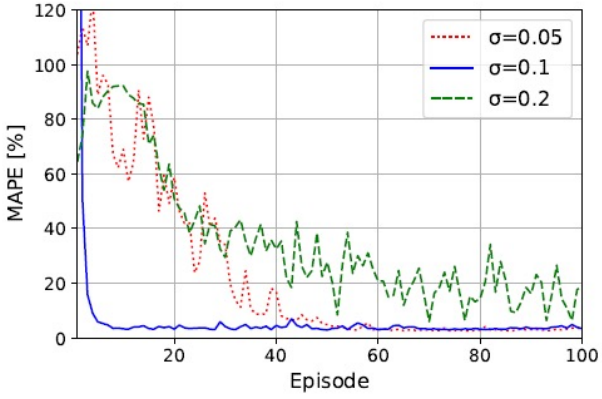
Fig. S-2. MAPE of the validation set to see the impact of hyperparameters on the performance of PPO (Solar dataset, $E_{\max} = 0.3$ p.u.).



(a) Hyperparameter for clipping (ϵ)



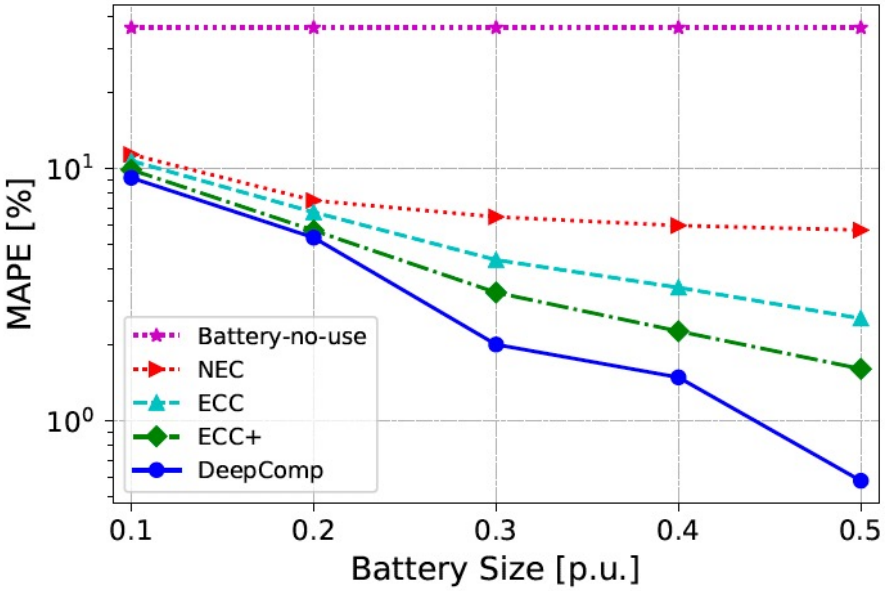
(b) Discount factor (γ)



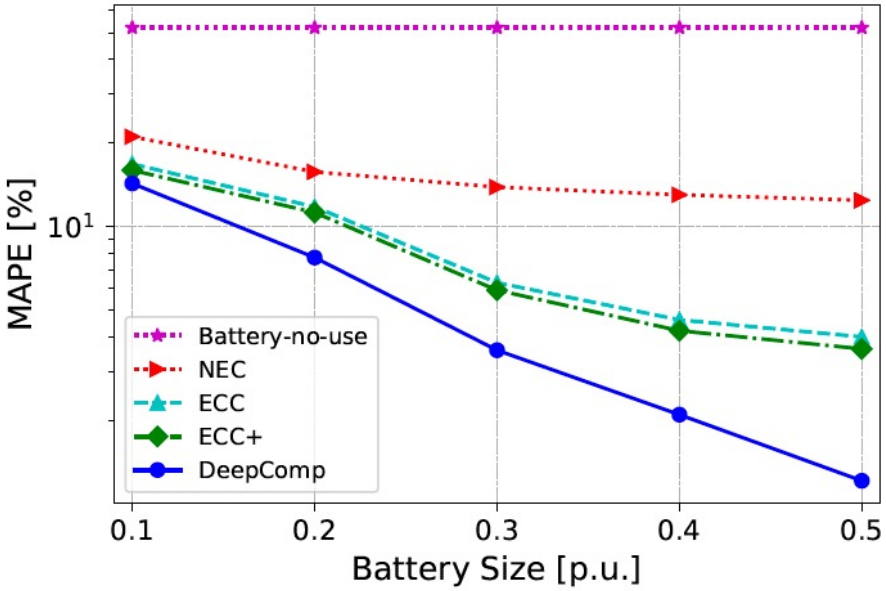
(c) Standard deviation (σ)

Fig. S-3. MAPE of the validation set to see the impact of hyperparameters on the performance of PPO (Wind dataset, $E_{\max} = 0.3$ p.u.).

MAPE Comparison



(a) Solar dataset



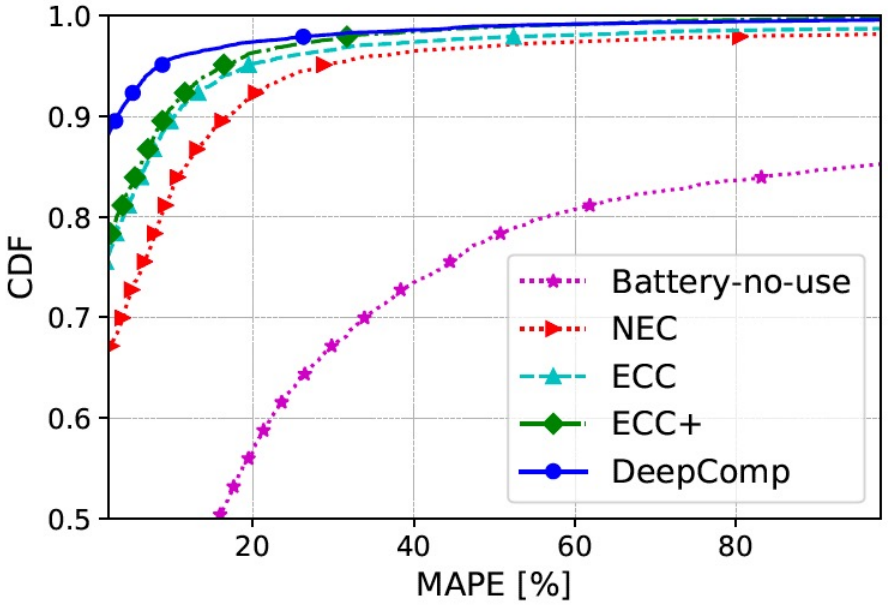
(b) Wind dataset

MAPE Comparison

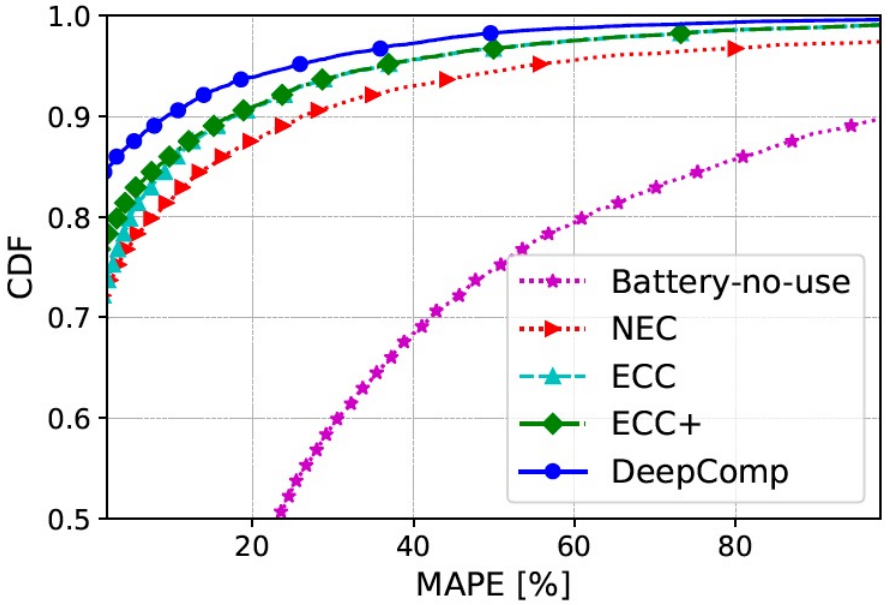
Metric	Model	Maximum battery capacity ($E_{\max}$)				
		0.1 p.u.	0.2 p.u.	0.3 p.u.	0.4 p.u.	0.5 p.u.
MAPE for solar [%]	Battery-no-use	36.16	36.16	36.16	36.16	36.16
	NEC	11.38	7.48	6.44	5.95	5.71
	ECC	10.74	6.72	4.35	3.38	2.55
	ECC+	9.90	5.69	3.23	2.27	1.61
	DeepComp	**9.20**	**5.33**	**2.01**	**1.49**	**0.58**
MAPE for wind [%]	Battery-no-use	52.10	52.10	52.10	52.10	52.10
	NEC	21.11	15.78	13.91	13.05	12.47
	ECC	16.85	11.81	6.30	4.63	4.02
	ECC+	15.99	11.27	5.92	4.23	3.64
	DeepComp	**14.34**	**7.76**	**3.60**	**2.11**	**1.22**

CDF Comparison

- DeepComp has highest value about 0.875 at MAPE=0, which implies that errors are completely compensated or 87.5% of time slots
- DeepComp is located distinctively upper left



(a) Solar dataset

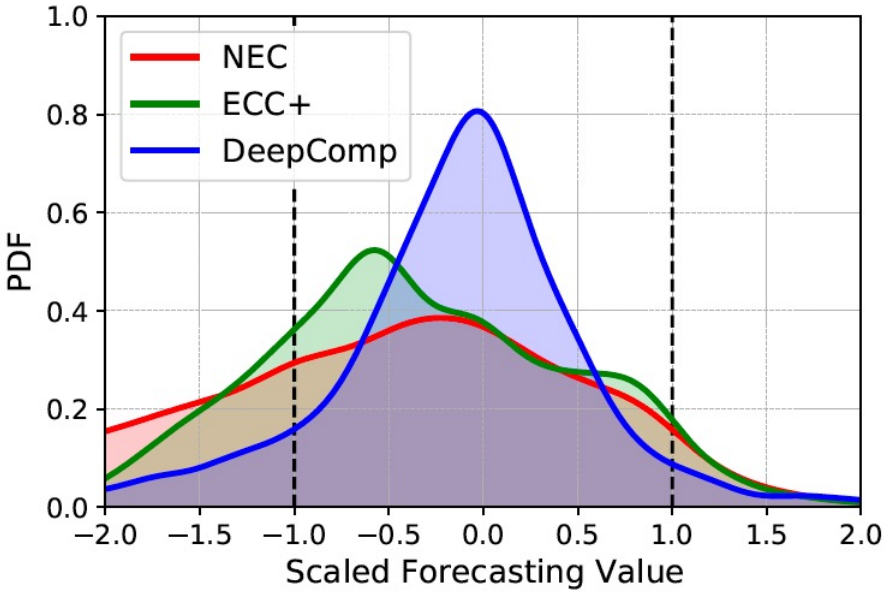


(b) Wind dataset

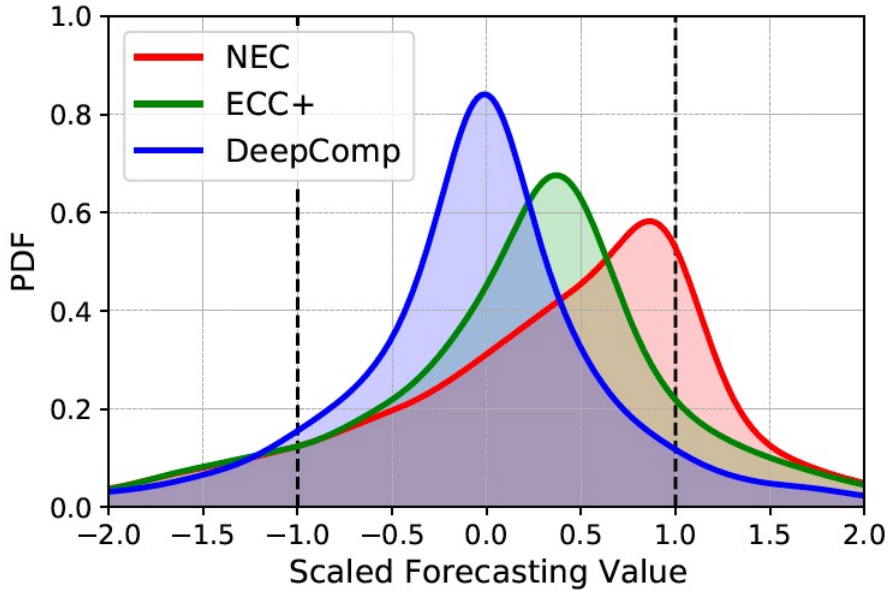
NICELAB
Networking for Intelligence Communications and Energy

PDF of Forecasting Values

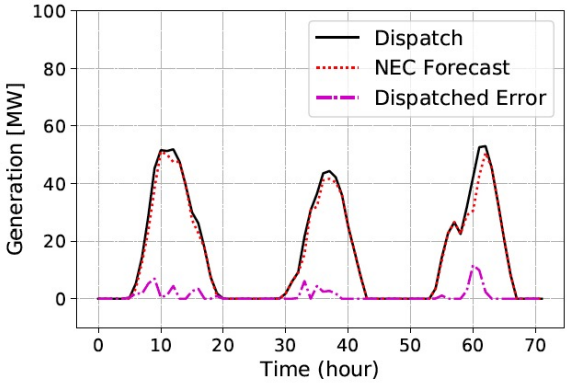
- Normalized by the error compensable value



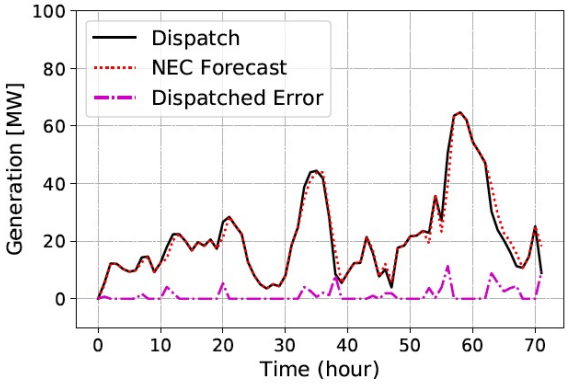
(a) Solar dataset



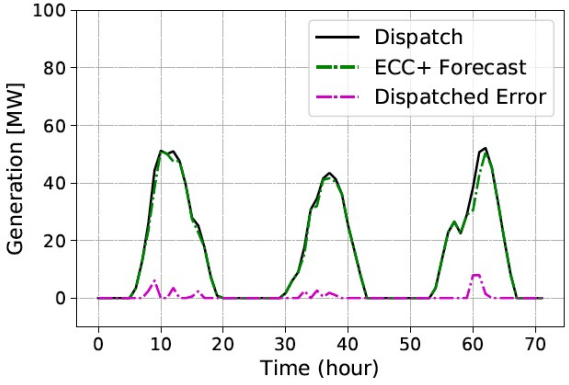
(b) Wind dataset



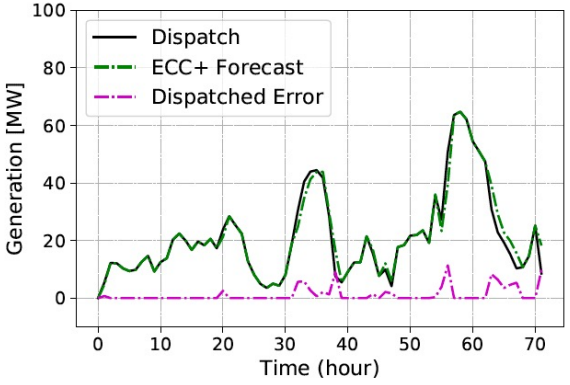
(a) NEC (solar)



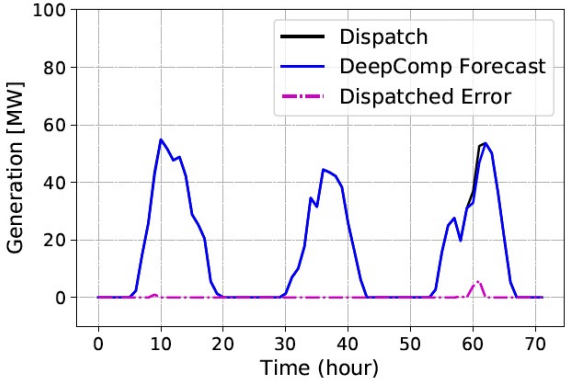
(b) NEC (wind)



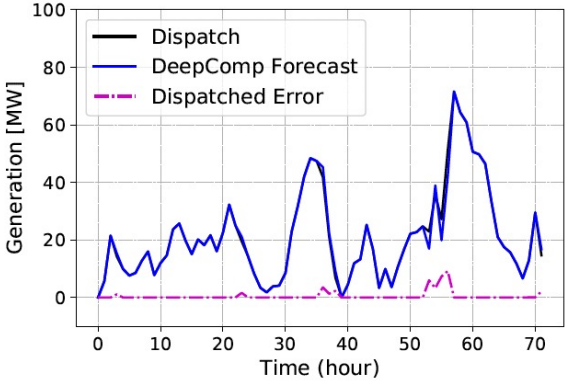
(c) ECC+ (solar)



(d) ECC+ (wind)



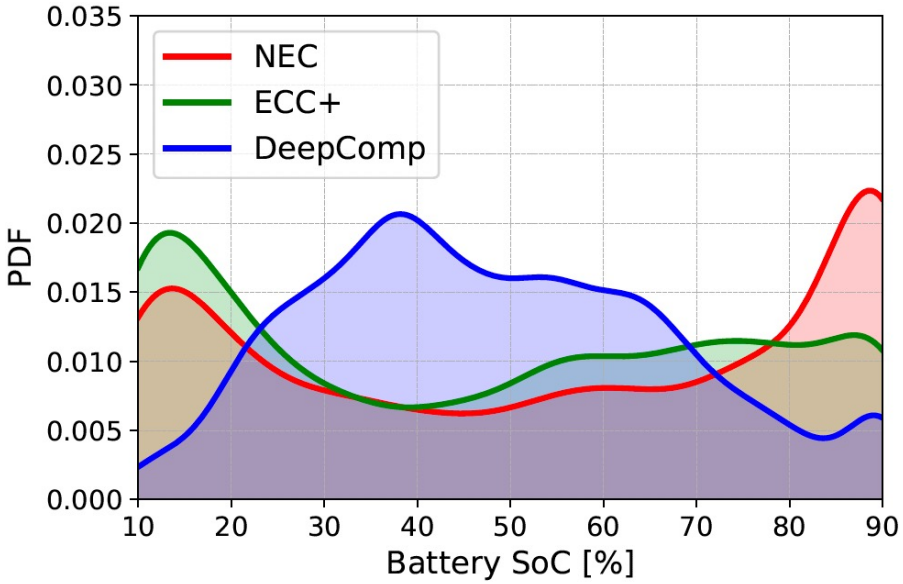
(e) DeepComp (solar)



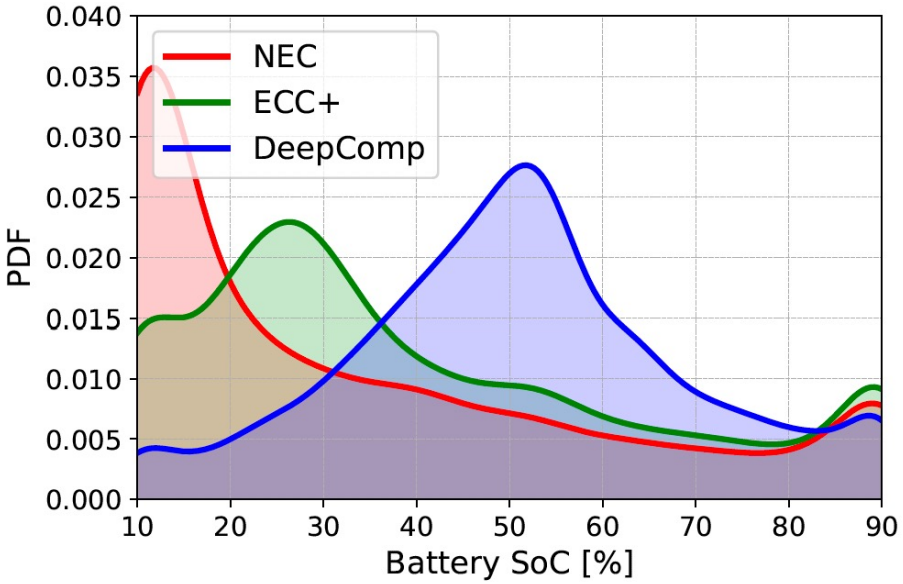
(f) DeepComp (wind)

Experimental Results

☐ PDF of the battery SoC



(a) Solar dataset



(b) Wind dataset

Conclusion

- We proposed a novel forecasting strategy called DeepComp
  - The objective is switched from reducing errors to making errors compensable
- We tackle the problem by leveraging DRL
  - The stored energy is affected by the previous forecasting results
- The proposed DeepComp shows significantly better performance than the other forecasting strategies
  - The MAPE is reduced by up to 90%
- In future work, we can generalize our one-step ahead forecasting into multi-step ahead forecasting

Networking
Next

Intelligence
Innovative

Communications
Creative

Energy
Envisioning

NICELAB

Networking for Intelligence Communications and Energy