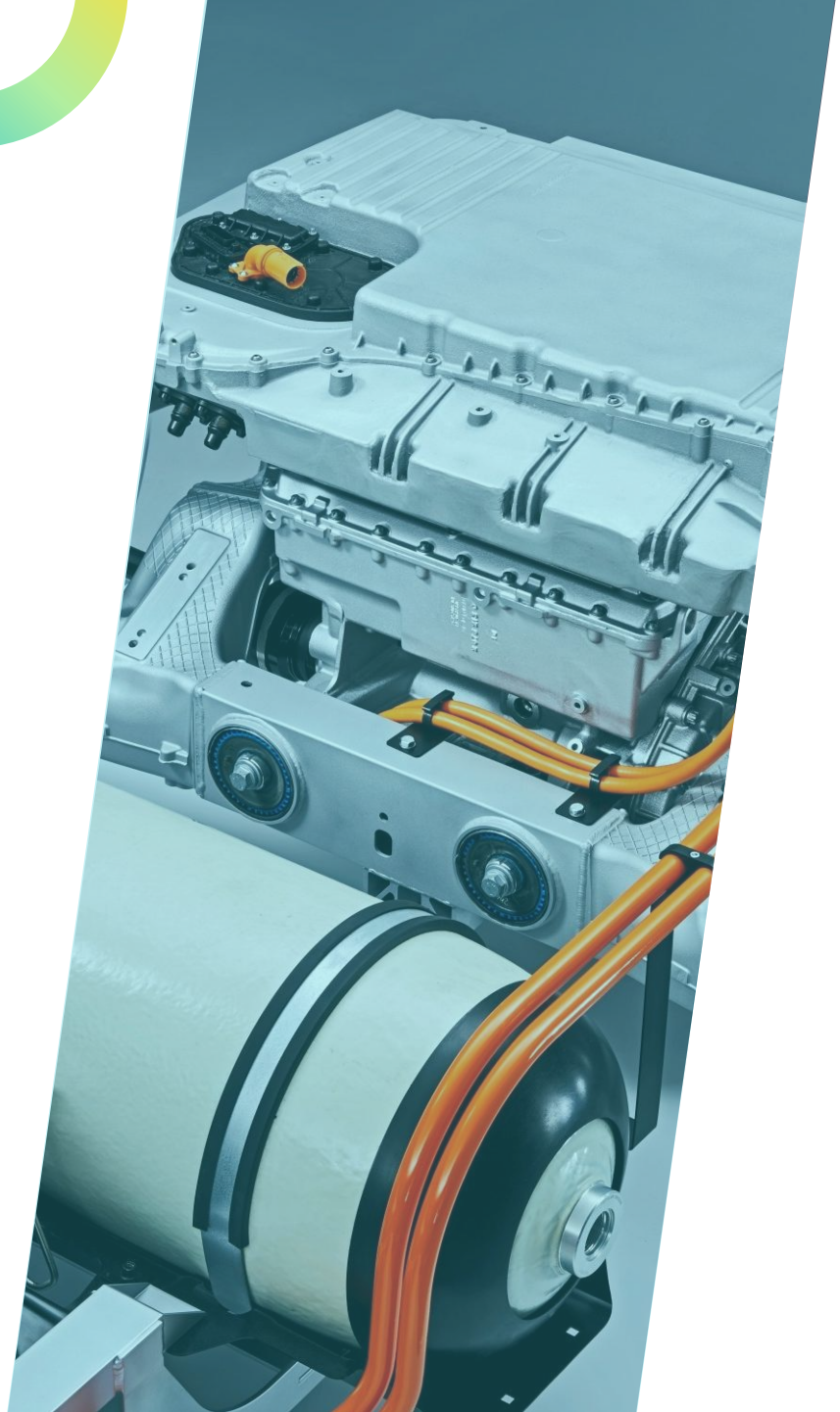
Beta-Policy SAC 기반,
수소전기차 에너지관리 전략의 비판적 분석 및 개선 연구

[강화학습 프로젝트]

2025.12.

학과 : 데이터사이언스 인공지능 학과
연구자 : 전현진 (A72078)

GitHub : https://github.com/ImNiceDS/FCHEV-SAC-Temperature-Penalty



I. 프로젝트 개요 및 연구 목적

1-1. (논문 리뷰) Beta-Policy SAC 기반 수소전기차 에너지관리 전략 연구

본 연구는 SAC 기반 강화학습을 사용하여 연료전지-배터리 하이브리드 시스템에서 Action : 연료전지 출력 비율(P_fcs), Reward : 에너지 소모, 연료전지·배터리 열화율, SOC 유지 비용을 최소화, 그 결과 경제성과 내구성을 동시에 향상시키는 최적 파워 분배 정책을 도출한다.

Energy Conversion and Management 292 (2023) 117362

Contents lists available at ScienceDirect

Energy Conversion and Management

journal homepage: www.elsevier.com/locate/enconman

Health-considered energy management strategy for fuel cell hybrid electric vehicle based on improved soft actor critic algorithm adopted with Beta policy

Weiqi Chen [a], Jiankun Peng [a,*], Jun Chen [a], Jiaxuan Zhou [a], Zhongbao Wei [b], Chunye Ma [a]

[a] School of Transportation, Southeast University, Nanjing 211102, China
[b] School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081, China

ARTICLE INFO

Keywords:
Fuel cell hybrid electric vehicle
State of health
Soft actor critic
Beta policy
Multi-objective optimization

ABSTRACT

Deep reinforcement learning-based energy management strategy (EMS) is essential for fuel cell hybrid electric vehicles to reduce hydrogen consumption, improve health performance and maintain charge. This is a complex nonlinear constrained optimization problem. In order to solve the problem of high bias caused by the inconsistency between the infinite support of stochastic policy and the bounded physics constraints of application scenarios, this paper proposes the Beta policy to improve standard soft actor critic (SAC) algorithm. This work takes hydrogen consumption, health degradation of both fuel cell system and power battery, and charge margin into consideration to design an EMS based on the improved SAC algorithm. Specifically, an appropriate tradeoff between money cost during driving and charge margin is firstly determined. Then, optimization performance differences between the Beta policy and the standard Gaussian policy are presented. Thirdly, ablation experiments of health constraints are conducted to show the validity of health management. Finally, comparison experiments indicate that, the proposed strategy has a 5.12% performance gap with dynamic programming-based EMS with respect to money cost, but is 4.72% better regarding to equivalent hydrogen consumption. Moreover, similar performances in validation cycle demonstrate good adaptability of the proposed EMS.

1. Introduction

The traditional transportation sector contributes approximately 20% of global greenhouse gas emissions and air pollution [1], which is a heavy burden on environment protection and energy security. Automobile companies and research institutes have been working to develop new vehicles to replace conventional internal combustion engine vehicles. At present, there are three mainstream technologies [2]: hybrid electric vehicles (HEV), fuel cell electric vehicles and pure electric vehicles. The HEV are only transitional product because they cannot avoid the high emissions and pollution of the internal combustion engine. Although pure electric vehicles have no emissions and pollution, they suffer from short driving range and long charging time due to the limitation of power battery technology [3].

In recent years, fuel cells have attracted more and more attentions because of high efficiency, no pollution, fast-fueling and low noise [4,5]. However, fuel cells have the disadvantages of slow dynamic response and poor stability in fast power demand conditions [6]. In order to ensure the sustainability of output power, a power battery with high energy density is generally equipped to be used together with fuel cells as an auxiliary energy source [7]. The power battery pack provides peak power to smooth fluctuations of fuel cells' output power [8]. However, hybrid energy storage makes the power and energy flow of the vehicle more complicated, so it is of great significance to formulate efficient and reasonable energy management and optimization strategies, so as to give full play to the performance and advantages of fuel cell hybrid electric vehicles (FCHEV).

As a fundamental and key technology of FCHEV, energy management strategy (EMS) is designed to distribute energy between fuel cell system (FCS) and power battery to improve powertrain efficiency and fuel economy [9], and also reduce health degradation of fuel cell and power battery [10]. The reported EMS can be classified as three categories [11]: 1) rule-based, 2) optimization-based, 3) learning-based.

The rule-based EMSs have the advantages of simplicity, easy implementation, low computation burden, and good reproducibility [12]. However, the rules are highly dependent on engineering

* Corresponding author.
E-mail address: jkpeng@seu.edu.cn (J. Peng).

https://doi.org/10.1016/j.enconman.2023.117362
Received 5 May 2023; Received in revised form 29 June 2023; Accepted 30 June 2023
Available online 5 July 2023
0196-8904/

<수소전기차 하이브리드 시스템 구성>

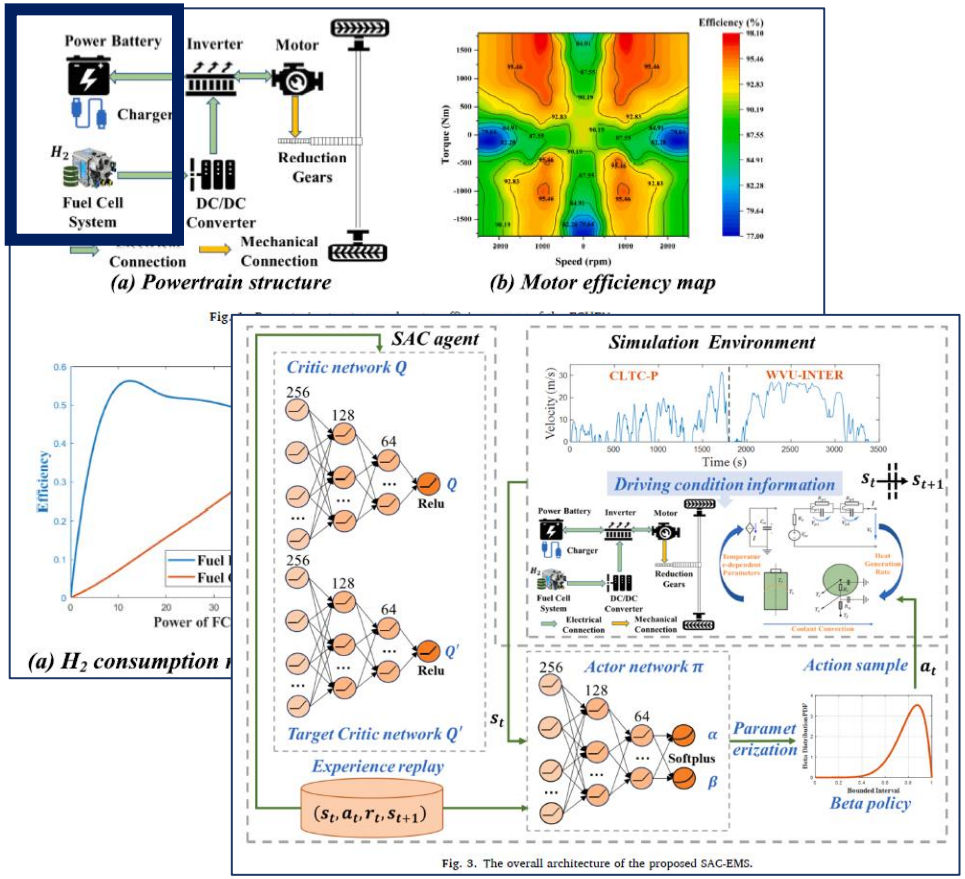


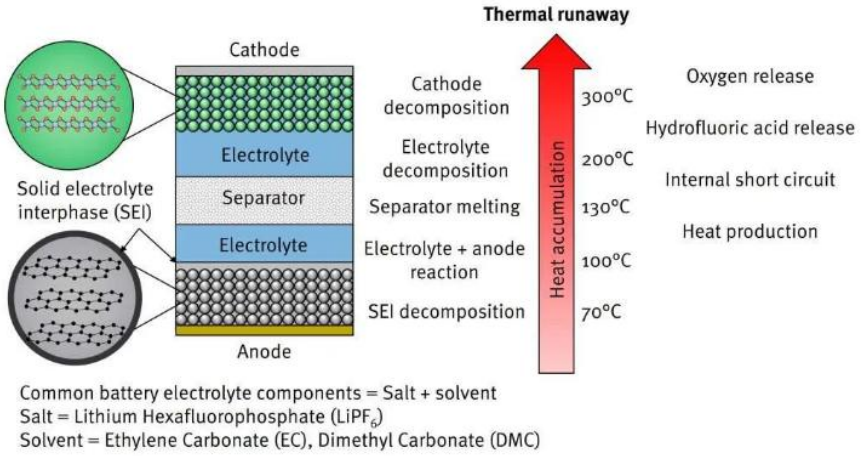
Fig. 3. The overall architecture of the proposed SAC-EMS.

I. 프로젝트 개요 및 연구 목적

1-2. 연구 목적 – 수소전기차 기술 전문성 기반 연구 결과의 비판적 분석 및 코드 개선 연구

ONLY 에너지 사용량 저감, 열화 저감을 위한 하이브리드 제어 전략의 문제점
**** 에너지 소모(수소 사용)를 줄이기 위해 배터리만을 과사용(과방전/과충전) 할 경우 배터리 열폭주 위험 ****

- 온도에 따른 배터리 열폭주 위험성



Battery Thermal Runaway - Battery Design

열폭주 위험의 온도 구간

- 약 **70 ~ 90 °C**: 내부 구성물(예: SEI 막)이 불안정해지기 시작 → 자체 발열, 내부 화학 반응 가속 가능성 있음. Hydrogen Journal +1
- **약 120 ~ 150 °C**: 분리막(Separator) 또는 전극/전해질 구조가 녹거나 분해되기 시작할 수 있는 단계. 이때부터 전극 간 단락(short)이나 급격한 발열 가능성이 커진다. Bonnen Battery +1
- 이후 급속 발열 → 셀 내부 화학 반응이 통제 불가하게 증가 → **폭주 단계 (thermal runaway)** 돌입. 이 때 온도는 **수백 °C → 800 °C 이상**까지 치솟을 수 있음. metisengineerin... +1

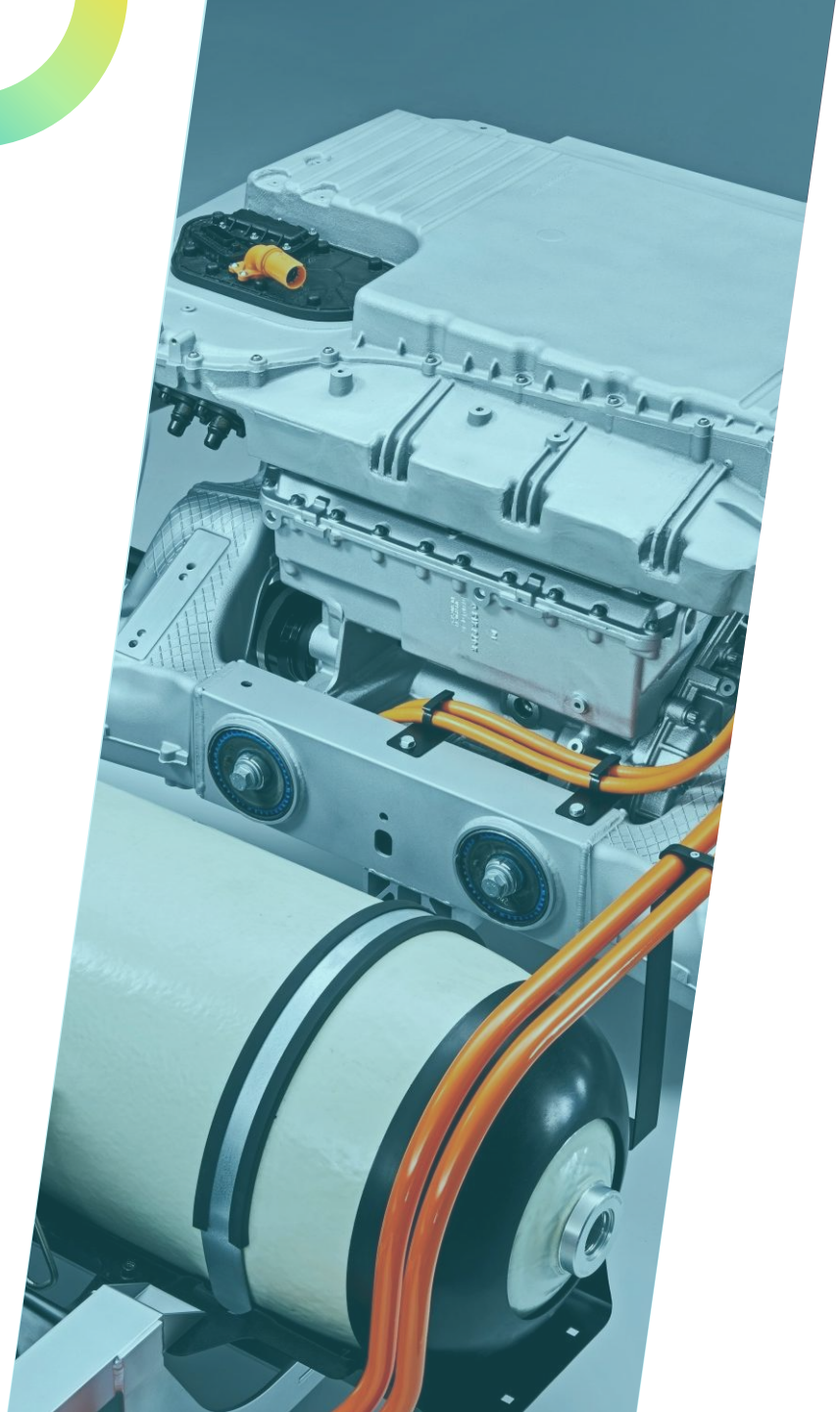
- 논문 구현 코드의 문제점 = Failure 조건

Only 배터리 SOC 조건만 관리

```
# electric model
SOC_new = SOC-soc_deriv
if SOC_new >= 1:
    Voc_new = self.ocv_func(1.0)  # for a cell#
    fail = True
    # SOC_new = 1.0
elif SOC_new <= 0.01:
    Voc_new = self.ocv_func(0.01)
    fail = True
    # SOC_new = 0.001
else:
    Voc_new = self.ocv_func(SOC_new)  # for a cell
    fail = False
# print('SOC: %.6f'%SOC_new)
```

논문의 Failure 조건 문제점 및 개선 요약

- 기존 논문은 배터리 **SOC 조건만을 Failure 기준**으로 사용하여, 실제 운용에서 가장 중요한 열 안전 요소가 반영되지 않는 한계가 존재함.
- 그러나 리튬이온 배터리는 **약 70°C 이상에서 SEI 붕괴, 내부 반응 가속, 자가발열 증가 등 열폭주 초기 단계 위험**이 발생할 수 있음.
- 온도 제한 없이 학습을 진행하면, 실제 차량에서는 절대 허용될 수 없는 **고온·고위험 운전 패턴도 정상 전략으로 학습되는 문제**가 존재함.
- 따라서 본 연구에서는 실제 강화학습 동안의 배터리 온도를 모니터링하여 위험성을 분석하고, **배터리 코어 온도(Tep_c ≥ 70°C)를 추가 Failure 조건으로 반영하여 현실성과 안전_**



01/ 프로젝트 개요 및 연구 목적

02/ **환경 및 데이터셋 (논문)**
2-1. 구현 코드
2-2. State / Action / Reward
3-3. 알고리즘 및 Hyperparameter
3-4. 실험 재현

03/ 강화학습 설계
3-1. 실험 변경 조건

04/ 실험 결과 및 분석

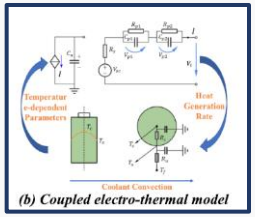
05/ 토의 및 결론

II. 환경 및 데이터셋

2-1. 코드 리뷰

**본 구현은 데이터가 아닌 물리 모델 기반으로 차량을 가상 환경에서 주행시켜,
env.py·agentEMS.py에서 EMS 환경을 구성하고 이를 통해 SAC가 연료전지 출력 비율(P_fcs)을 학습한다.**

```
[ main.py ]
    |
    ├─ 불러오기 → [ arguments.py ]
    |
    ├─ 환경 생성 → [ env.py ]
    |                  |
    |                  ├─ 주행 사이클 입력 (CLTC-P, WVUINTER ...)
    |                  └─ 동력 요구 계산(P_dem)
    |
    ├─ EMS 제어 → [ agentEMS.py ]
    |                  |
    |                  ├─ 상태(state) 구성
    |                  ├─ 보상(reward) 계산
    |                  └─ 물리 모델 실행:
    |                                |
    |                                ├─ [ Cell.py ] 배터리 (전기·열·열화)
    |                                ├─ [ FCHEV_SOH.py ] 연료전지 수명
    |                                ├─ [ MotorModel ] 모터
    |                                └─ [ dcdcModel ] DC/DC 변환
    |
    ├─ RL 학습 → [ sac.py ] / [ network.py ] / [ memory.py ]
    |
    └─ 반복 관리 → [ runner.py ]
```



(b) Coupled electro-thermal model

**데이터가 아닌 물리모델 기반으로
차량이 가상조건에서 운행됨**

1. 실행 파트 (main / runner)

- `main.py`에서 전체 실험을 실행하며 학습 파라미터를 로드한다.
- `runner.py`가 에피소드 반복, RL 학습 호출, 데이터 저장을 담당한다.
  → **전체 실험을 조율하는 상위 제어부**

2. 환경 파트 (env / utils)

- 주행 사이클(CLTC-P, WVUINTER 등)을 불러와 차량의 속도·가속도 프로파일을 구성한다.
- 매 step마다 요구동력(P_dem)을 계산하고, RL의 action을 물리모델에 전달한다.
  → **RL이 학습하는 '가상 도로 환경'**

3. 차량 제어기 파트 (agentEMS)

- 상태(state: SOC, 온도, SOH 등)를 생성하고 RL의 action(연료전지 출력)을 적용한다.
- 수소 소비·열화·SOC 유지 등을 고려해 보상(reward)을 계산한다.
  → **에너지 관리 전략(EMS)의 핵심 로직**

4. 강화학습 알고리즘 파트 (sac / network / memory)

- Soft Actor-Critic(SAC) 알고리즘과 Beta-Policy Actor를 구현한다.
- Critic(Q-network)와 Replay Buffer를 이용해 안정적인 학습을 수행한다.
  → **최적 EMS 정책을 학습하는 RL 엔진**

5. 물리모델 파트 (Cell / FCHEV_SOH / Motor / DCDC)

- 배터리(전기·열·수명), 연료전지(효율·열화), 모터·DC/DC 효율 등
  차량 구성요소의 실제 물리 반응을 계산한다.
  → **RL이 제어하는 '가상 차량 모델'**

Energy Conversion and Management 292 (2023) 117362

II. 환경 및 데이터셋

2-2. State / Action / Reward

에이전트는 [State] 차량·배터리·연료전지의 상태를 관측하여, [Action] 연료전지 출력(P_fcs)을 결정하고, [Reward] 에너지 소비·열화(SOH)·SOC 유지 비용을 최소화하는 보상으로 최적 EMS 전략을 학습한다.

① State 구성 (총 7개 변수)

강화학습 에이전트는 FCHEV의 에너지 흐름·건강 상태를 나타내는 아래 7개의 상태를 관측한다.

상태 변수	의미	범위	출처
SOC	배터리 State of Charge	[0, 1]	배터리 전기 모델
SOH_bat	배터리 열화 상태	[0, 1]	배터리 열·열화 모델
SOH_fcs	연료전지 성능 열화 상태	[0, 1]	FCS degradation 모델
P_bat	배터리 출력(충·방전값)	−200 ~ 200 kW	차량 특성
P_fcs	연료전지 출력	0 ~ 60 kW	차량 특성
v	차량 속도	0 ~ 120 km/h	주행 사이클
a	차량 종방향 가속도	−3 ~ 2 m/s²	주행 사이클

🔎 **해석:**

- SOC·SOH는 장·단기 성능 모두 반영
- FCS와 배터리 양쪽의 열화·출력·열 조건이 모두 포함
- 속도·가속도는 요구 파워(P_dem)를 결정하는 핵심 주행 조건
- 이를 통해 **학습 에이전트는 실시간 차량 운행·에너지 흐름·열화 상태를 모두 고려한 의사결정이 가능해짐**

② Action 정의

Action = 연료전지 출력(P_fcs)

- SAC 에이전트의 행동은 **연료전지 출력 비율**, 즉 FC가 어느 정도 출력을 낼지 결정하는 것
- Action 범위는 0~60 kW이며, 실제 논문에서는 **Beta Policy 기반 bounded action**으로 출력
- P_fcs가 결정되면 나머지는 아래처럼 자동 결정됨:
  - 요구 파워 P_dem이 주어짐
  - P_bat = P_dem − P_fcs
  - 배터리 전기·열 모델, DC/DC 모델, 모터 모델이 연쇄적으로 실행되어 시스템 상태가 업데이트됨

🔎 **해석:**

- Action을 단일화(P_fcs)함으로써 학습 안정성 향상
- 배터리/연료전지 간 power split을 implicit하게 자동 조절
- 하이브리드 시스템의 핵심 제어변수가 연료전지 출력이기 때문에 매우 합리적인 설계

③ Reward 설계

논문에서는 연료전지·배터리의 경제성· 내구성·SOC 유지라는 3가지 목표를 **금전 비용(Driving Cost)**으로 통합하여 보상으로 사용함.

Reward 공식

$$r_t = -\left[\rho_1 \dot{m}(t) + \rho_2 D_{fcs}(t) + \rho_3 \Delta SOH(t) + \omega |SOC(t) - SOC_{ref}|\right]$$

항목별 의미

항목	설명
ρ1 · ṁ(t)	수소 소비량(연료전지 연료비)
ρ2 · D_fcs(t)	연료전지 열화에 따른 비용 증가
ρ3 · ΔSOH(t)	배터리 열화(SOH 감소)에 따른 비용 증가
**ω·	SOC−SOC_ref

II. 환경 및 데이터셋

2-3. 알고리즘 및 Hyperparameter

**Beta Policy가 적용된 개선형 SAC 알고리즘을 사용해 연료전지 출력 제어를 학습하며,
안정적 수렴을 위해 최적의 신경망 구조·학습률·버퍼·배치 크기 등의 하이퍼파라미터를 설정한다.**

① 사용 알고리즘: Improved SAC (Soft Actor–Critic with Beta Policy)

본 연구는 연속 제어 문제에 적합한 **Soft Actor–Critic(SAC)**을 사용하며,
특히 기존 Gaussian Policy의 무한 범위 문제를 해결하기 위해 **Beta Policy**를 적용한 개선된 SAC를 사용한다.

🔹 **Improved SAC의 핵심 특징**

1. **Maximum Entropy Framework 적용**
   - 보상 + 정책의 엔트로피를 동시에 최대화
   - 탐색 성능 향상 및 안정적인 학습
2. **Clipped Double Q-Network**
   - 2개의 Q-Network로 과대평가 문제(overestimation) 감소
3. **Value Network + Target Value Network**
   - Soft Bellman Backup을 사용하여 값 함수 안정화
4. **Beta Policy (Gaussian → Beta 분포로 개선)**
   - Action 범위가 물리적으로 0~1(또는 P_fcs의 범위)로 제한되므로
     Beta 분포가 물리 제약과 일치해 **bias-free**
   - Gaussian 사용 시 발생하면 boundary truncation bias 제거
   - **수렴 속도 개선 + 행동 품질 향상**
     (논문 Figure 7(a)에서 84 episodes vs 224 episodes)

② 알고리즘 동작 과정 (요약)

1. **현재 상태 s_t 입력**
2. **Actor가 Beta Policy를 통해 연료전지 출력(P_fcs) 샘플링**
3. **환경 실행 → 다음 상태 s_{t+1}, 보상 r_t 획득**
4. **Replay Buffer에 transition 저장 (s, a, r, s')**
5. **미니배치 샘플링 후 네트워크 업데이트**
   - Q-Network: Soft Bellman residual 최소화
   - Value Network: 일정한 soft value 타깃에 맞춰 학습
   - Policy Network: KL divergence 기반 업데이트
6. **Target Value Network는 soft update 적용 (Polyak τ=0.001)**

③ Hyperparameter 설정 (논문 Table 3 기반)

하이퍼파라미터	값	설명
Actor/Critic hidden layers	256 → 256	2-layer MLP
Optimizer	Adam	모든 네트워크 공통
Learning rate (Actor)	**5e-5 ~ 5e-3 (Cyclical)**	주기적 변화로 local minima 탈출
Learning rate (Critic)	**5e-5 ~ 5e-4 (Cyclical)**	안정적 학습
Discount factor (γ)	0.99	장기 horizon 반영
Target soft update (τ)	0.001	Polyak averaging
Replay buffer size	4×10^4	경험 재사용
Batch size	64	샘플링 학습
Training episodes	400	Mix-train 기준
Policy distribution	**Beta(α,β)**	bounded action 정책
Action range	0~60 kW	P_fcs의 물리적 한계

④ 논문 기반 Improved SAC의 장점

- Gaussian → Beta Policy 개선으로 **수렴속도 2.6배 빨라짐**
- FCS·배터리 열화를 반영하는 **multi-objective 에너지 관리 최적화**
- DP 대비 **경제성 5.12% gap, 수소 효율 4.72% 우수**
- 다른 DRL (DQN·DDPG) 대비 **SOC 유지·SOH 안정성 우수**

II. 환경 및 데이터셋

2-4. 논문 재현

논문에서 제시한 SAC-EMS를 동일 조건으로 재현한 결과,
본 실험은 연비(Equivalent H_2)와 비용(Driving Cost) 모두 논문 대비 동등하거나 더 우수한 성능을 확인하였다.

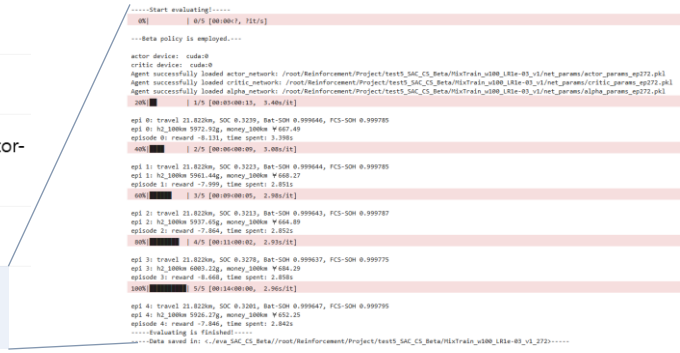
Table 6
Comparison of different EMSs.

Method	Equivalent H_2 cost (g/100 km)	Comparison of H_2	Driving cost (CNY/100 km)	Comparison of money
DP	6454.1	100%	710.37	100%
DQN	6504.1	99.23%	1137.52	62.45%
DDPG	6663.4	96.86%	777.13	91.41%
SAC	6163.2	104.72%	748.73	94.88%

Energy Conversion and Management 292 (2023) 117362

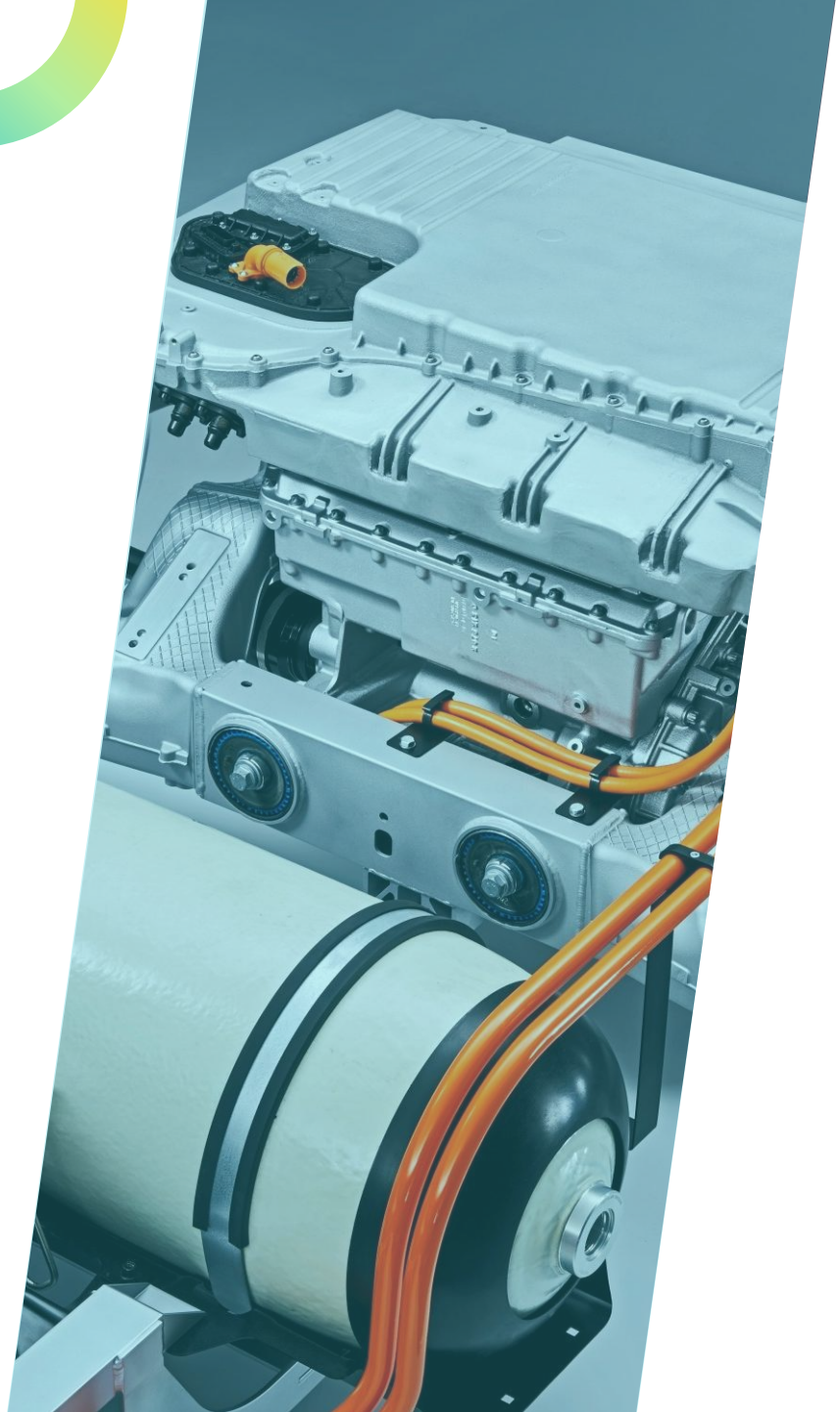
📊 논문 4개 메서드 + JEON 재현 결과 비교표

Method	Equivalent H_2 (g/100 km)	DP 대비 (%)	Driving Cost (CNY/100 km)	DP 대비 (%)	비고
DP (논문)	**6454.1**	**100%**	**710.37**	**100%**	기준선
DQN (논문)	6504.1	99.23%	1137.52	62.45%	값 기반 RL
DDPG (논문)	6663.4	96.86%	777.13	91.41%	Deterministic Actor-Critic
SAC (논문)	**6163.2**	**104.72%**	**748.73**	**94.88%**	Beta Policy SAC
SAC (재현 평가, JEON)	**≈ 5960 g**	**108.3%**	**≈ 667 RMB**	**106.5%**	MixValid 평가



→ 논문 동일 조건 재현 결과, 동등 or 더 우수한 성능 확인

→ 실제 학습 / 평가 결과



III. 강화학습 설계

3-1. 기존 코드 문제점

본 논문 코드는 배터리 SOC만을 Failure 조건으로 고려해 코어 온도 기반 안전 한계를 전혀 반영하지 못하며, 이로 인해 고온에 따른 배터리 열폭주 가능성을 무시한 비현실적 EMS 제어 전략을 학습하게 된다.

- 온도에 따른 배터리 열폭주 위험성

📈 **열폭주 위험의 온도 구간**

- 약 **70 ~ 90 °C**: 내부 구성물(예: SEI 막)이 불안정해지기 시작 → 자체 발열, 내부 화학 반응 가속 가능성 있음. Hydrogen Journal +1
- **약 120 ~ 150 °C**: 분리막(Separator) 또는 전극/전해질 구조가 녹거나 분해되기 시작할 수 있는 단계. 이때부터 전극 간 단락(short)이나 급격한 발열 가능성이 커진다. Bonnen Battery +1
- 이후 급속 발열 → 셀 내부 화학 반응이 통제 불가하게 증가 → **폭주 단계 (thermal runaway)** 돌입. 이 때 온도는 **수백 °C → 800 °C 이상**까지 치솟을 수 있음. metisengineerin... +1

- 논문 코드의 문제점

① Failure 조건 부재(온도 미반영)

SOC만 고려되어 고온 상황에서도 episode가 종료되지 않아 **배터리 열폭주 위험을 무시한 비현실적 학습이 발생함.**

② 배터리 온도 강제 클리핑

Tep_a가 60°C로 고정되어 **실제 고온 상태와 열화 특성이 시뮬레이션에서 사라짐.**

③ 온도 정보 미기록(out_info 누락)

코어 온도 기록이 없어 **고온 failure의 원인 분석 및 안전성 디버깅이 불가능함.**

① Only 배터리 SOC 조건만 관리

```
# electric model
SOC_new = SOC-soc_deriv
if SOC_new >= 1:
    Voc_new = self.ocv_func(1.0)  # for a cell#
    fail = True
    # SOC_new = 1.0
elif SOC_new <= 0.01:
    Voc_new = self.ocv_func(0.01)
    fail = True
    # SOC_new = 0.001
else:
    Voc_new = self.ocv_func(SOC_new)  # for a cell
    fail = False
# print('SOC: %.6f'%SOC_new)

Voc_new = Voc_new*13.87/168
V1_new = V1+v1_deriv
V2_new = V2+v2_deriv

# terminal voltage
Vt_new = Voc_new+V1_new+V2_new+self.r0*I_batt
power_out = Vt_new*I_batt
```

② 배터리 평균온도 강제 클리핑

```
# thermal model
Tep_c_new = Tep_c+tc_deriv
Tep_s_new = Tep_s+ts_deriv
Tep_a_new = (Tep_c_new+Tep_s_new)/2
if Tep_a_new > 60:
    Tep_a_new = 60

# aging model
Bc = self.Bc_func(Ic_rate)
E = 31700-370.3*Ic_rate
# T = Tep_a_new+273.15
T = 25 + 273.15   # constant temperature assumption
Ah = (20/Bc/math.exp(-E/8.31/T))**(1/0.55)  # z = 0.55, ideal_gas_constant = 8.31
N1 = 3600*Ah/self.Cn
dsoh = self.timestep*(abs(I_batt/2/N1/self.Cn))
SOH_new = SOH-dsoh
```

③ 배터리 코어온도 관리x

```
out_info = {'SOC': SOC_new, 'SOH': SOH_new, 'soc_deriv': soc_deriv,
            'cell_OCV': Voc_new, 'cell_Vt': Vt_new, 'cell_V_3': V_3,
            'cell_V1': V1_new, 'cell_V2': V2_new,
            'I': I_batt, 'I_c': Ic_rate, 'cell_power_out': power_out,
            'P_batt': P_batt/1000, 'tep_a': Tep_a_new, 'dsoh': dsoh}
paras_list_new = [SOC_new, SOH_new, Tep_c_new, Tep_s_new, Tep_a_new, Voc_new, V1_new, V2_new]
done = fail
return paras_list_new, dsoh, I_batt, done, out_info
```

III. 강화학습 설계

3-1. 기존 코드 문제점

기존 코드의 Reward 구조는 단일 SOC 기반 보상 구조로,
SOC를 정상 범위에 유지하도록 하고, Fail 임계구간(≥0.95/≤0.05) 접근 시 강한 패널티 부여

- 기존 코드 - Reward 구조

```
def get_reward(self):
    # equivalent hydrogen consumption
    if self.P_batt > 0:
        h2_batt = self.P_batt / 1000 * 0.0164      # g in one second
        # 取FCS效率最高点(0.5622)计算, 该系数为0.0164
    else:
        h2_batt = 0
    h2_equal = self.h2_fcs + h2_batt

    if self.SOC >= 0.95 or self.SOC <= 0.05:
        w_soc = self.w_soc * 10     # 不可直接改变self.w_soc的值
    else:
        w_soc = self.w_soc
    soc_cost = w_soc * abs(self.SOC - self.SOC_target)
```

✅ Reward 재설계 (기존 방식 vs 개선 방식)

1) 기존 Reward 구조 (SOC 제어 중심 + Fail 조건 패널티)

■ 핵심 아이디어

- ✔ SOC를 적정 범위(≈0.5)에서 유지하도록 유도
- ✔ SOC가 Fail 임계구간(≥0.95 또는 ≤0.05)에 접근하면 큰 패널티 부여
- ✔ 즉, **SOC 안정성 유지가 Reward 설계의 유일한 안전기준**

■ 동작 로직 설명

- SOC가 정상 범위이면 일반적인 SOC 오차 비용 적용
- SOC가 0.95 이상 또는 0.05 이하로 접근할 경우
  → 패널티 가중치(w_soc)를 **10배로 증가**
  → 강화학습 에이전트가 즉시 SOC 회복 행동을 선택하도록 유도
- Fail 상태에 대한 직접적 제재만 존재하며
  배터리 온도(Tep_c) 등 물리적 안전성과는 무관

> **→ 한계점:**
> 실제 차량에서 가장 위험한 요소인 *배터리 온도 상승(열화 가속, 열폭주 위험)*을 고려하지 않음.
> SOC 제어 하나만으로는 장기적인 안전성과 성능을 보장하기 어려움.

III. 강화학습 설계

3-2. 실험 변경 조건 ** 코드 개선 ① **

기존 코드의 온도 미고려·로그 부족 문제를 개선하기 위해 배터리 코어 온도 기반 Failure 조건을 추가하고 배터리 코어 온도 값을 로그에 기록하여 안전성·현실성을 갖춘 학습 환경으로 수정하였다

- 코드 개선 내용

```
# 처음에 한 번만 초기화
fail = False

if SOC_new >= 1:
    Voc_new = self.ocv_func(1.0)  # for a cell#
    fail = True
elif SOC_new <= 0.01:
    Voc_new = self.ocv_func(0.01)
    fail = True
else:
    Voc_new = self.ocv_func(SOC_new)  # for a cell
    # 여기서는 fail을 건드리지 않음

# 🔥 여기서 온도 fail 추가
if Tep_c_new >= 60:
    fail = True
# print('SOC: %.6f'%SOC_new)

Voc_new = Voc_new*13.87/168
V1_new = V1+v1_deriv
V2_new = V2+v2_deriv

# terminal voltage
Vt_new = Voc_new+V1_new+V2_new+self.r0*I_batt
power_out = Vt_new*I_batt

# aging model
Bc = self.Bc_func(Ic_rate)
E = 31700-370.3*Ic_rate
# T = Tep_a_new+273.15
T = 25 + 273.15   # constant temperature assumption
Ah = (20/Bc/math.exp(-E/8.31/T))**(1/0.55)  # z = 0.55, ideal_gas_constant = 8.31
N1 = 3600*Ah/self.Cn
dsoh = self.timestep*(abs(I_batt/2/N1/self.Cn))
SOH_new = SOH-dsoh

out_info = {'SOC': SOC_new, 'SOH': SOH_new, 'soc_deriv': soc_deriv,
            'cell_OCV': Voc_new, 'cell_Vt': Vt_new, 'cell_V_3': V_3,
            'cell_V1': V1_new, 'cell_V2': V2_new,
            'I': I_batt, 'I_c': Ic_rate, 'cell_power_out': power_out,
            'P_batt': P_batt/1000, 'tep_a': Tep_a_new,'tep_c': Tep_c_new, 'dsoh': dsoh}
paras_list_new = [SOC_new, SOH_new, Tep_c_new, Tep_s_new, Tep_a_new, Voc_new, V1_new, V2_new]
done = fail                                    # ★★★★★★★★★★★★★★
return paras_list_new, dsoh, I_batt, done, out_info
```

① Failure 조건 추가
배터리 코어 온도(Tep_c_new) ≥ 60℃

② 배터리 코어 온도(Tep_c_new) 저장 로그 추가

🔧 코드 개선 사항

1. Failure 조건의 현실성 부족 문제 개선

기존 모델은 **SOC 과충전(SOC ≥ 1.0) / 과방전(SOC ≤ 0.01)** 만을 Failure로 처리하고 있었으며,
배터리 온도(특히 코어 온도)에 대한 어떤 안전 제약도 존재하지 않았다.

그러나 실제 배터리는 **코어 온도(Tep_c)** 가 안전 운용의 가장 중요한 요소이며,
70 °C 이상에서는 정상 동작이 불가능하다.

따라서 다음과 같이 Failure 조건을 강화하였다.

- **배터리 코어 온도(Tep_c_new ≥ 70 °C)** 시 즉시 Failure 처리
- 고온 상태에서의 비현실적인 충·방전 동작을 학습하지 않도록 안전 제약을 명확히 반영
- SOC 기반만으로 이루어진 기존 Failure 구조의 물리적 비현실성을 개선

이 수정으로 강화학습 에이전트는 온도 상승을 제어해야 한다는 제약을 제대로 학습하게 된다.

---

2. 시뮬레이션 로그(out_info) 정보 부족 문제 개선

기존 out_info에는 **Tep_a(평균 온도)** 만 기록되었으며,
심지어 평균 온도는 코드 내부에서 **60 °C로 클리핑되어 실제 값을 반영하지 못했다.**

결과적으로:

- 고온 Failure의 원인을 분석할 수 없음
- 실제 배터리 내부 열 상태를 기반으로 한 후처리·검증 불가능
- 온도 기반 안전성 분석 및 논문 재현성에 큰 결함 존재

이를 해결하기 위해 다음을 개선하였다.

- **배터리 코어 온도(Tep_c_new)** 를 out_info에 새로 추가 저장
- clipping 되지 않은 실제 코어 온도로 후처리 분석 가능
- Failure 원인(SOC vs 온도)을 명확히 추적 가능
- 온도 기반 정책학습의 타당성을 정량적으로 검증 가능

III. 강화학습 설계

3-2. 실험 변경 조건 ** 코드 개선 ② **

**개선된 Reward 구조는 배터리 코어 온도(Tep_c)를 추가로 고려하며,
58°C 초과 시 온도 초과량에 비례한 강한 패널티를 부여하여 고온 운행을 억제함으로써,
보다 안전하고 현실적인 에너지 관리 전략으로 확장됨**

- 개선 코드 - Reward 구조

```
# battery temperature cost
w_temp = 200.0
if self.Tep_c > 58:
    temp_cost = w_temp * (self.Tep_c - 58)
else:
    temp_cost = 0
```

① 배터리 코어 온도 ≥ 58℃ 조건 큰 패널티 부여

② temp_cost 추가

```
reward = -(money_cost + soc_cost + temp_cost)
reward = float(reward)
self.info.update({'EMS_reward': reward, 'soc_cost': soc_cost,
                  'h2_equal': h2_equal, 'h2_batt': h2_batt,
                  'money_cost': money_cost, 'h2_money': h2_money,
                  'batt_money': batt_money, 'fcs_money': fcs_money,
                  'money_cost_real': money_cost_real,
                  'temp_cost': temp_cost})
return reward
```

2) 개선 Reward 구조 (배터리 코어 온도 기반 패널티 추가)

■ 핵심 개선 포인트

- ✔ SOC 안정성 외에 **배터리 코어 온도(Tep_c)**를 직접 감시
- ✔ Tep_c > 58°C 도달 시 **즉각적이고 큰 패널티 부여**
- ✔ 온도 상승 억제를 통해 실제 배터리 열화 및 안전성을 반영한 RL 학습 구조 완성

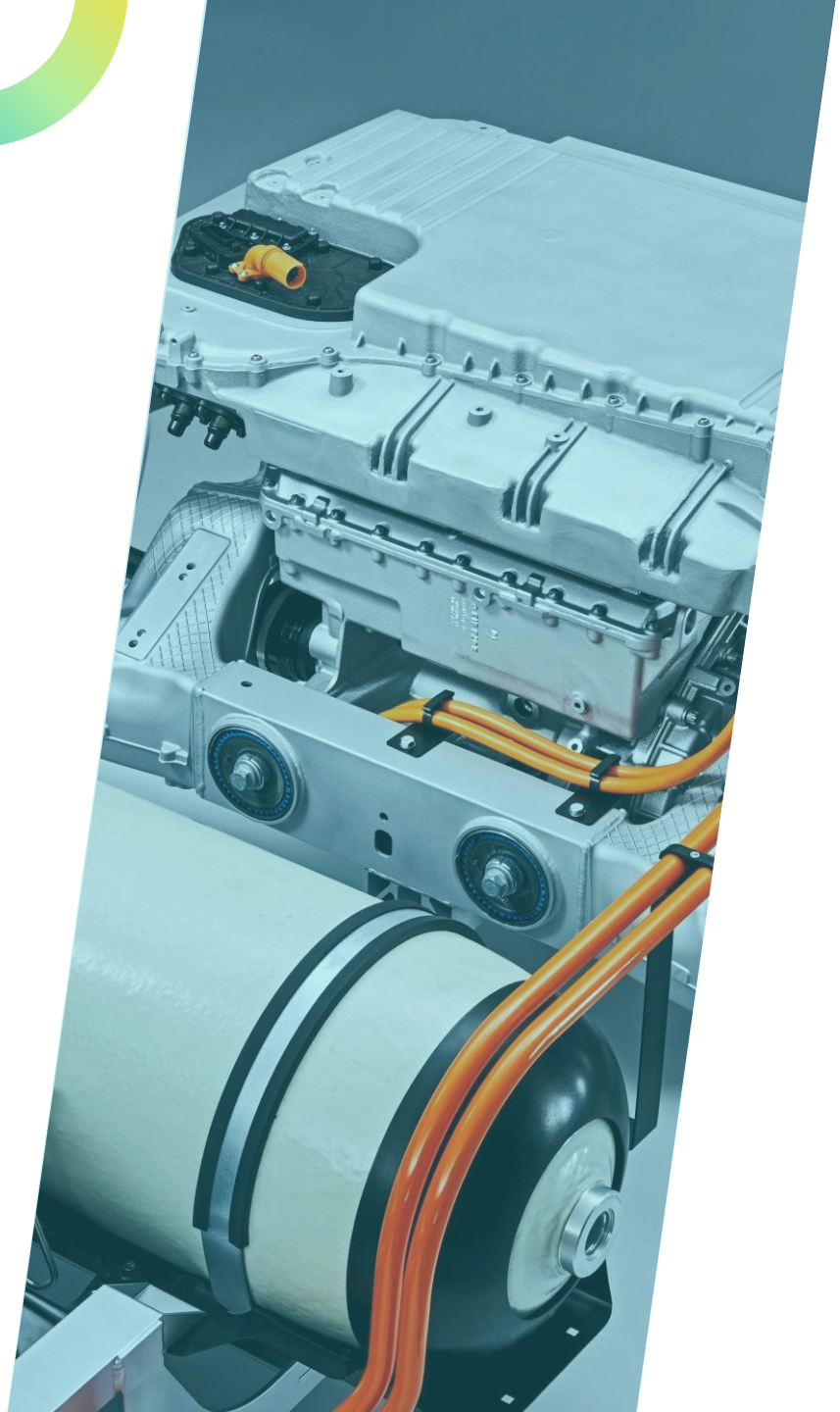
■ 설계 원칙

- Tep_c는 배터리 SOH 저하 및 안전성 문제를 야기하는 핵심 인자
- 58°C 이상 구간은 열화 가속이 크게 증가하는 위험 구간
- 따라서 보상에서 *강한 불이익*을 부여하도록 설계

■ 동작 로직 설명

- Tep_c ≤ 58°C
  → 온도 패널티 없음 (정상 범위)
- Tep_c > 58°C
  → 온도 초과량(℃) × w_temp 만큼 패널티 부여
  → w_temp = 200으로 설정하여 **즉각적으로 행동을 교정하도록 강한 신호 제공**

> **→ 기대 효과:**
> - "배터리 온도 상승 억제"
> - "고온 상태에서의 급격한 열화 방지"
> - "고 C-rate 충/방전 억제로 SOH 장기 유지"
> - "전체 에너지 분배 정책(P_fcs/P_batt)이 부드럽고 안정적이 되는 효과"



IV. 실험 결과 및 분석

4-1. 실험 결과 및 분석

보상 조건은 변경하지 않고, 배터리 코어 온도의 모니터링만 추가하여 검증한 결과,
논문에서 학습한 정책은 대부분의 에피소드에서 배터리 코어 온도가 60°C를 초과하는 고온 상태를 반복하였다.
이는 논문의 Beta-Policy SAC 기반 전략이 핵심 안전 조건을 누락한 채 에너지 소비만 최소화하도록 학습함.

✅ 배터리 코어 온도 관찰 결과 및 문제점 정리

본 연구에서는 기존 논문의 에너지 관리 전략의 한계를 확인하기 위해,
Failure 조건(배터리 코어 온도 ≥ 70°C)만 추가하고 reward 구조는 수정하지 않은 상태에서
학습 중 배터리 코어 온도 변화를 분석하였다.

1) Failure 조건만 추가한 학습 결과

- 배터리 코어 온도 Fail 조건(≥70°C)을 포함했음에도,
  reward에 온도·안전성 요소가 없기 때문에 정책은 온도 위험을 회피하지 못함.
- Fail은 episode 종료 신호일 뿐이며 패널티가 없으므로,
  에이전트는 **고온 상태를 위험으로 인식하지 않음.**

2) 그래프 해석 — Max Tep_c per Episode

- 초기 에피소드에서 **75~78°C까지 급격히 상승**하며 즉시 Fail 발생.
- 이후에도 학습된 정책은 **60~70°C 고온 구간**에서 반복적으로 등작함.
- 배터리 안전 관점에서 **Tep_c 60°C 이상은 안정적 운전이 어려운 영역**이며,
  현실 차량에서는 사용 불가능한 수준의 열 조건임.

3) 결론: 기존 논문 EMS 전략의 구조적 문제

- 논문 기반 SAC-EMS는
  - 수소 소비 저감
  - 연료전지·배터리 열화 비용 저감
  - SOC 타깃 유지

  만을 학습 목표로 삼고 있으며,
  온도·열폭주 위험을 전혀 고려하지 않음.
- 그 결과, 학습된 정책은 **배터리를 지속적으로 고온 구간으로 몰아넣는 비안전적 제어 전략**을 형성함.

- 학습 결과

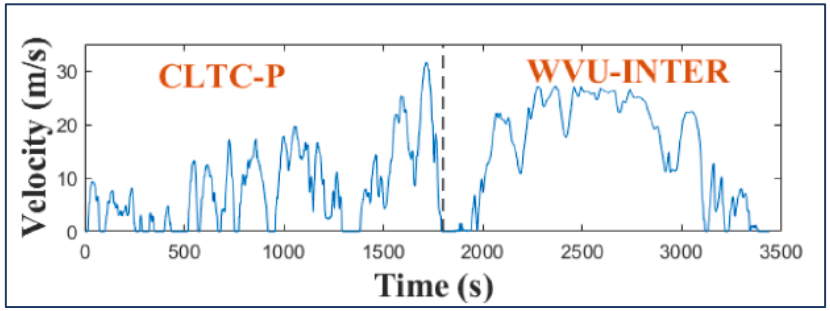

```
failure in step 2504 of episode 0
  1%|          | 1/100 [00:52<1:26:10, 52.22s/it]

epi 0: travel 39.438km, SOC -0.7410, FCS-SOH 0.999528, Bat-SOH 0.999018
epi 0: H2_100km 1961.4g, eq_H2_100km 5546.8g, money_100km ￥715.68
epi 0: ep_r -335.778, c-loss1 44723.7062, c-loss2 45782.1068, a-loss 255.1807, en-loss 0.0937
epi 0: lr_critic 0.000050, lr_actor 0.000050, lr_alpha 0.000050

failure in step 578 of episode 1
  2%|▏         | 2/100 [01:51<1:31:46, 56.19s/it]

epi 1: travel 39.438km, SOC 0.6710, FCS-SOH 0.999775, Bat-SOH 0.999260
epi 1: H2_100km 4834.0g, eq_H2_100km 7788.7g, money_100km ￥624.47
epi 1: ep_r -664.000, c-loss1 209214.1535, c-loss2 190267.6434, a-loss 2916.7588, en-loss -1.1930
epi 1: lr_critic 0.000145, lr_actor 0.000055, lr_alpha 0.000055

failure in step 1584 of episode 2
  3%|▎         | 3/100 [02:54<1:35:45, 59.23s/it]

epi 2: travel 39.438km, SOC -1.9250, FCS-SOH 0.999957, Bat-SOH 0.997978
epi 2: H2_100km 92.5g, eq_H2_100km 4334.9g, money_100km ￥550.73
epi 2: ep_r -845.280, c-loss1 217345.9041, c-loss2 163596.6566, a-loss 8041.7888, en-loss -3.0158
epi 2: lr_critic 0.000240, lr_actor 0.000060, lr_alpha 0.000060
```

IV. 실험 결과 및 분석

4-1. 실험 결과 및 분석 (Train – driving cycle 기준 비교)

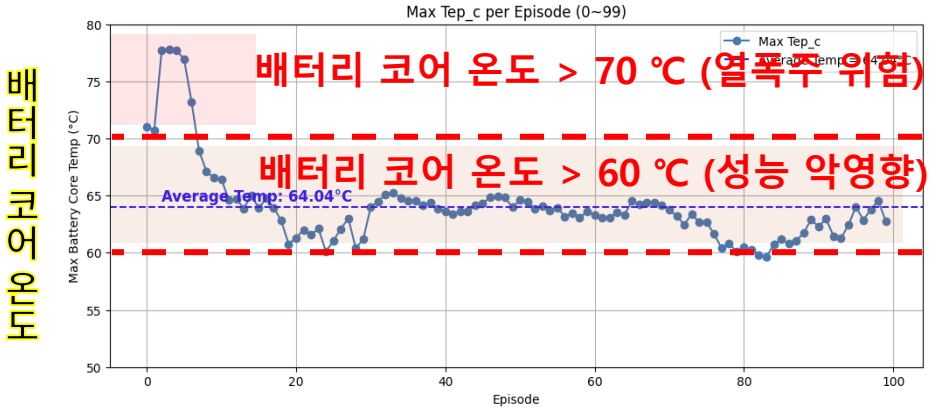
온도 보상 적용 이전에는 학습 조건에서 **배터리 코어 온도가 60~70℃까지 상승**하며 위험 구간에 자주 진입했으나, **개선 후**에는 **최대 온도가 55~60℃ 수준으로 안정화**되어 고온 영역 진입이 효과적으로 억제되었다.

- **Train** - Driving cycle

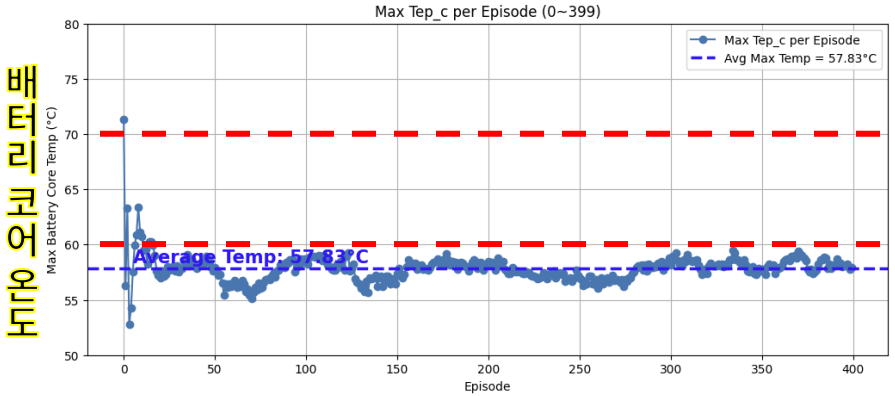


- 학습 결과

<논문 코드>



<개선 코드>

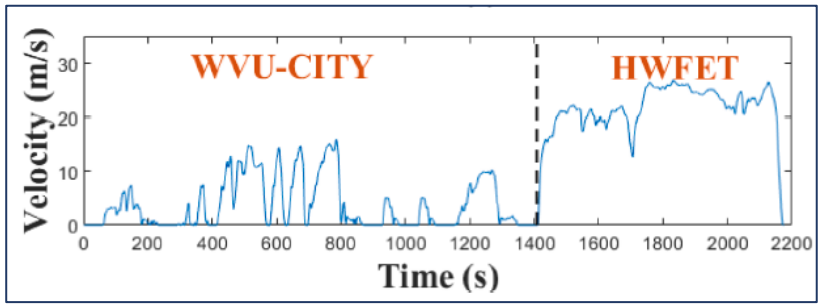


IV. 실험 결과 및 분석

4-1. 실험 결과 및 분석 (Valid – driving cycle 기준 평가 결과)

Valid driving cycle은 전체적으로 운전 부하가 낮아 기존 코드에서도 위험 온도(>58°C) 구간에 진입하지 않았으나, 보상 구조 개선 이후에는 배터리 코어 온도가 평균적으로 추가 저감되며 열적 안정성이 향상됨
반면, 온도 억제 전략으로 인해 Total Reward는 하락, 에너지 소모(Eq. H_2)는 소폭 증가하였음

- **Valid** - Driving cycle

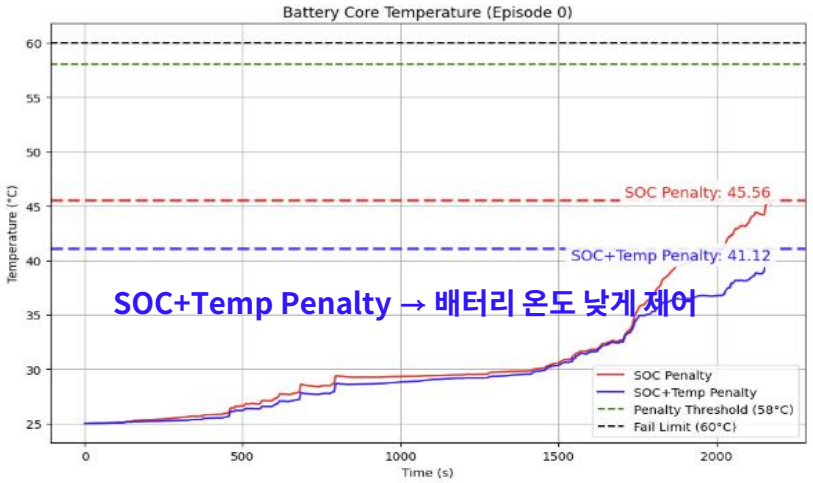


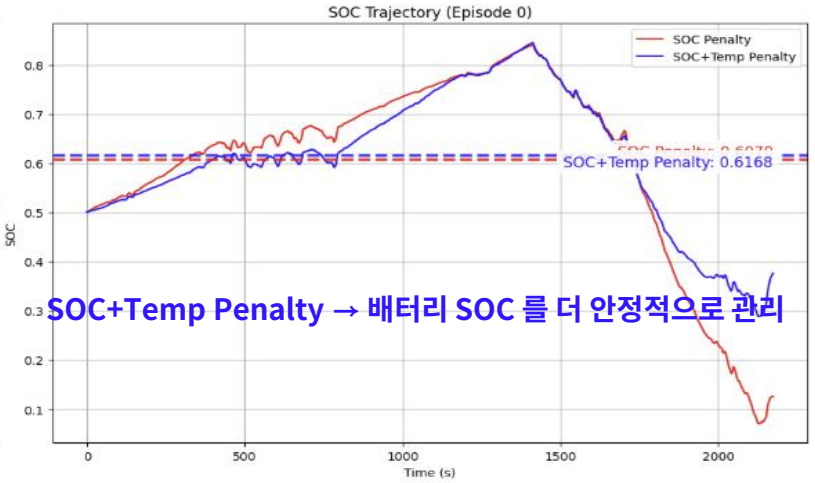
- 학습 결과

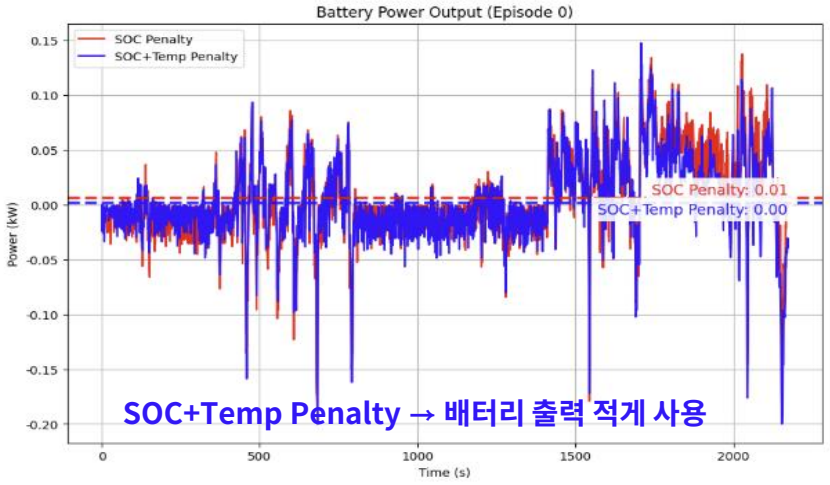
IV. 실험 결과 및 분석

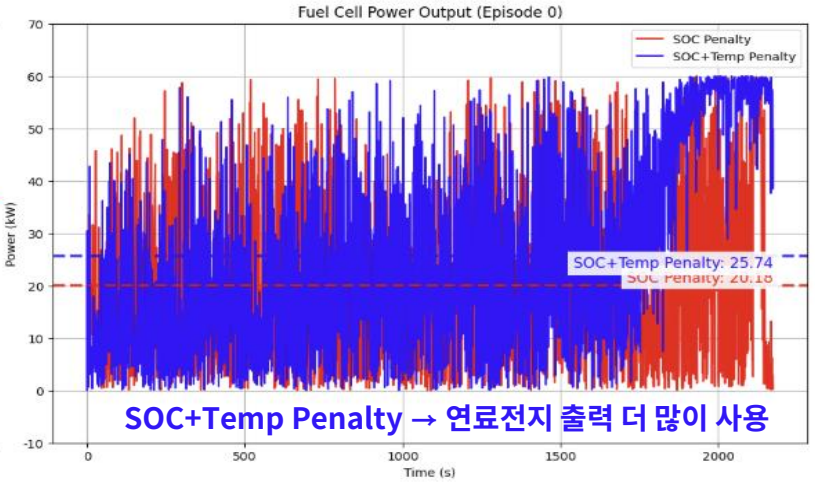
4-1. 실험 결과 및 분석 (Valid – driving cycle 기준 평가 결과)

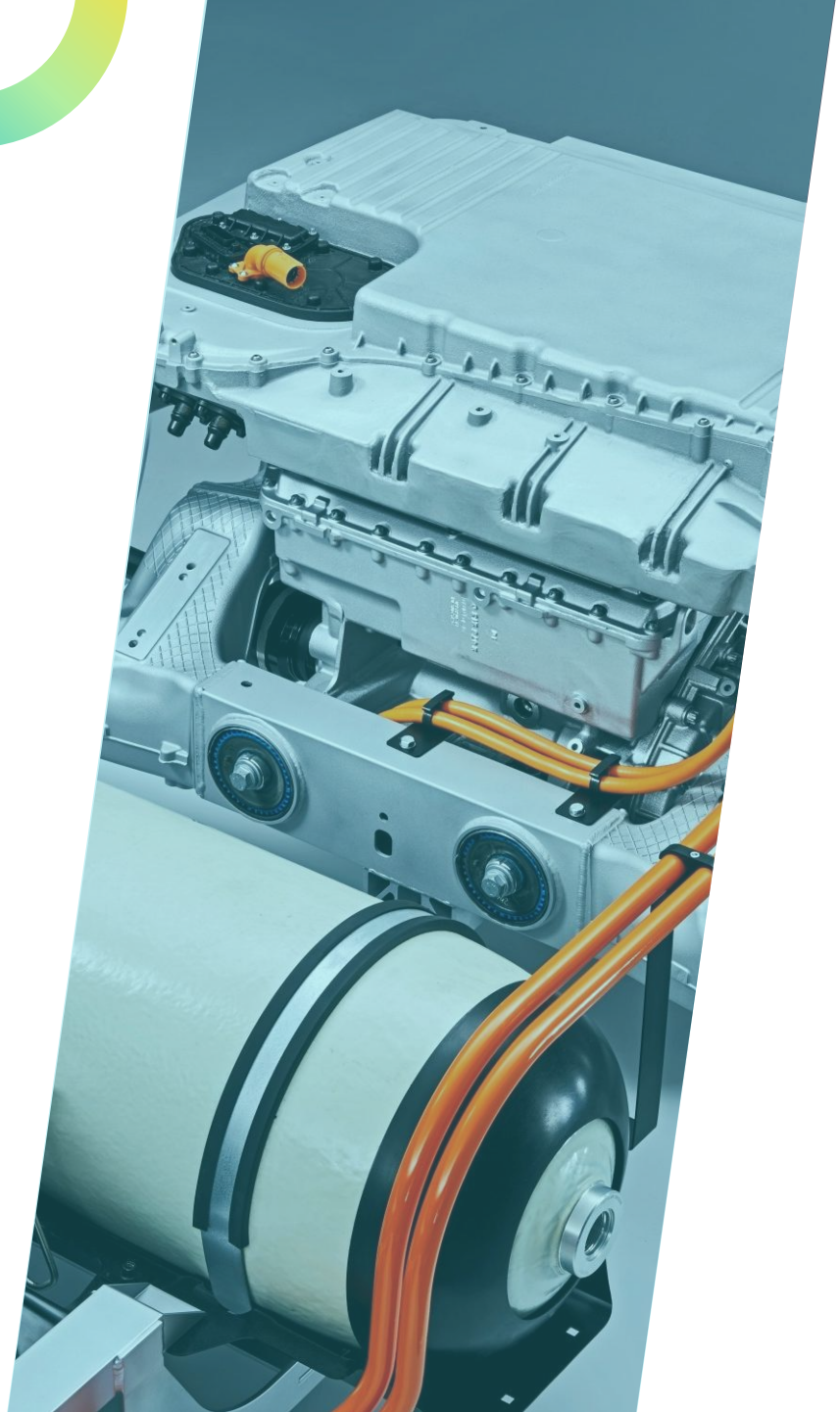
(데이터 심층 분석) SOC+Temperature 복합 보상 구조를 적용한 이후
배터리 SOC는 학습 전반에서 30% 이상을 안정적으로 유지하며 과방전 위험이 사라졌으며,
전체 파워트레인 관점에서는 더 안전하고 보수적인 운행 패턴으로 변화했음을 확인











GitHub : https://github.com/ImNiceDS/FCHEV-SAC-Temperature-Penalty

V. 토의 및 결론

5-1. 연구 기여도

**본 연구는 Beta-Policy SAC 기반 EMS의 실제 시스템 안전성을 검증하였으며,
특히, 핵심 안전 요소인 배터리 코어 온도에 대한 보상 패널티를 설계·추가 적용함으로써,
실제 차량 적용이 가능한 보다 현실적이고 안전 중심의 에너지 관리 전략으로 발전시켰음.**

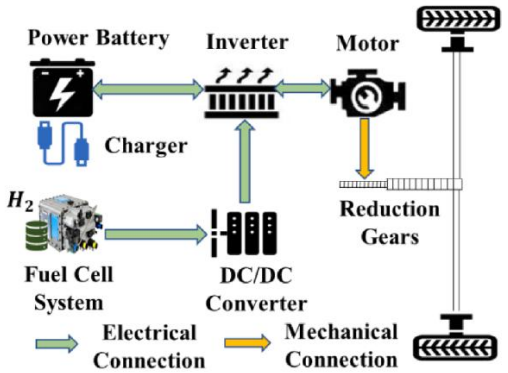
✅ 토의 (Discussion)

- 기존 SOC 기반 보상에서는 **Train cycle에서도 배터리 코어 온도가 약 64.0°C 수준으로 유지되었으나,**
  Temperature penalty 적용 후에는 **약 57.8°C로 저감**되어 열 안정성이 향상됨.
- 온도 억제를 위해 에이전트가 배터리 출력을 평균적으로 더 적게 사용하며,
  FCS 출력은 **20.2 kW → 25.7 kW** 수준으로 증가하는 보수적 에너지 분배 전략을 형성함.
- 전반적으로 안전성은 강화되었지만, 그 대가로 **Eq. H_2 소비량이 소폭 증가**하는 trade-off가 관찰됨.

✅ 결론 (Conclusion)

- 제안한 SOC + Temperature 복합 보상 구조는 기존 방식 대비
  배터리 열적 스트레스를 효과적으로 줄여 **더 안전한 운행 조건**을 달성함.
- 에너지원 사용 패턴도 현실적인 방향으로 조정되어, 실제 차량 적용 시
  배터리 수명 및 안전 확보에 유리한 전략임을 확인함.

*** 수소전기차 하이브리드 시스템**



감사합니다