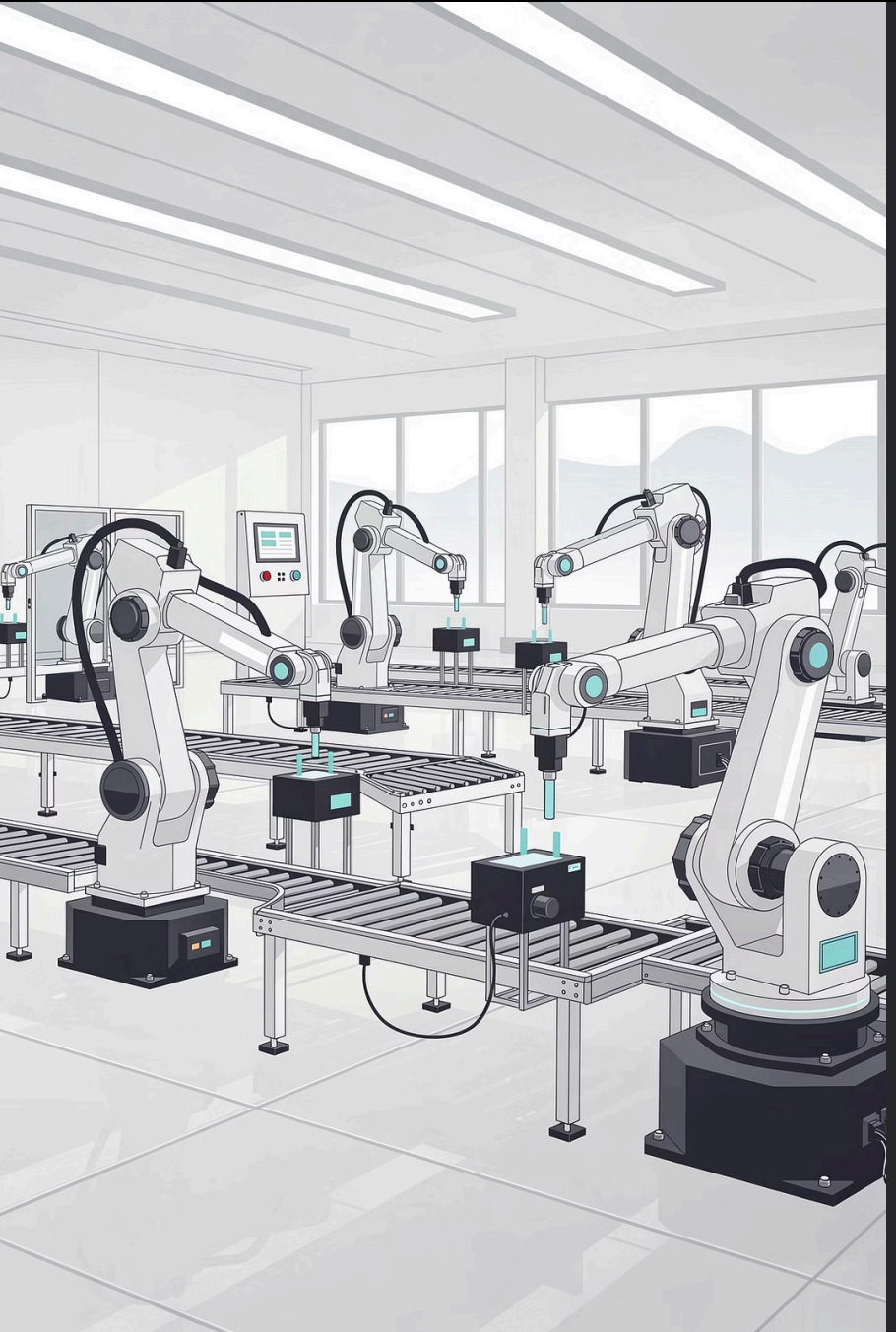


강화학습 기반
제조 설비 유지보수 스케줄링

AI4I 2020 Predictive Maintenance 데이터 기반 가상 시뮬레이터

A70033 김경훈

연구 배경 & 문제 정의

기존 정비 방식의 한계

제조 설비 고장은 생산 중단과 막대한 비용 손실을 초래함. 전통적인 정비 방식은 두 가지 극단 사이에서 딜레마에 직면해 있습니다.

- **과도한 예방 정비**: 불필요한 다운타임과 정비 비용 증가
- **부족한 예방 정비**: 예기치 않은 고장, 생산 중단, 불량품 발생

연구 목표

센서 데이터를 기반으로 최적의 정비 시점을 스스로 결정하는 강화학습(RL) 에이전트를 개발합니다.

핵심 목표: 생산 이익 극대화와 고장 리스크 최소화를 동시에 달성하는 장기적인 정책(Policy) 학습

프로젝트 전체 구조

1단계: 데이터 이해

AI4I 2020 데이터셋 분석 및 고장 규칙 파악

2단계: 시뮬레이터 설계

가상 설비 환경 구축 (State/Action/Reward/Failure)

3단계: RL 에이전트 학습

DQN 기반 강화학습 모델 설계 및 훈련

4단계: 성능 평가

Baseline 대비 성능 비교 및 분석

데이터셋 개요 – AI4I 2020

본 연구는 스마트 제조 환경을 모사한 **AI4I 2020 Predictive Maintenance Dataset**을 활용했습니다. 이는 물리 기반 규칙으로 생성된 synthetic 데이터로, 실제 예지보전 시나리오를 충실히 반영합니다.

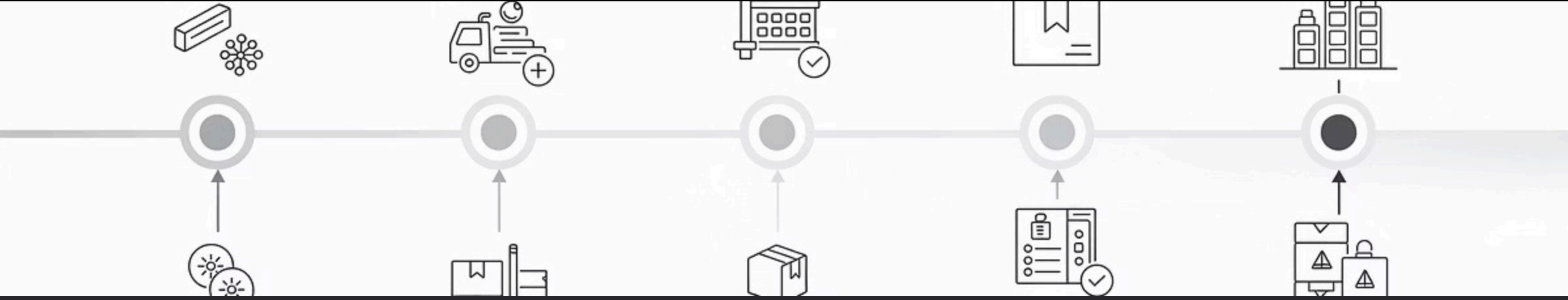
주요 상태 변수

- **Air Temperature [K]**: 주변 온도
- **Process Temperature [K]**: 공정 작동 온도
- **Rotational Speed [rpm]**: 모터 회전 속도
- **Torque [Nm]**: 부하 토크
- **Tool Wear [min]**: 공구 마모 누적 시간 (핵심 변수)

고장 모드 (5가지 유형)

- **TWF**: Tool Wear Failure (공구 마모 고장)
- **HDF**: Heat Dissipation Failure (열 발산 고장)
- **PWF**: Power Failure (전력 고장)
- **OSF**: Overstrain Failure (과부하 고장)
- **RNF**: Random Failure (무작위 고장)

데이터 특성: 정적 데이터셋을 시계열 특성을 가진 동적 강화학습 환경(Environment)으로 변환하여 사용



에피소드 & 타임스텝 개념

Time Step (타임스텝)

제품 1개를 가공하는 한 번의 생산 작업 단위입니다. 각 step마다 에이전트는 현재 상태를 관측하고 행동을 결정합니다.

Episode (에피소드)

설비를 새로 세팅한 후, 고장이 발생하거나 최대 T_max step에 도달할 때까지의 전체 운전 과정입니다.

학습 목표

장기 누적 보상 Σ r_t를 최대화하여, 전체 운영 기간 동안의 비용을 최소화하는 최적 정책을 학습합니다.

State(상태) 벡터 설계

상태 s_t는 현재 설비의 건강 상태, 공정 조건, 그리고 정비 이력을 종합적으로 표현합니다. 에이전트는 이 정보를 바탕으로 최적의 행동을 결정합니다.

온도 정보

- Air temperature
- Process temperature

기계 동작

- Rotational speed
- Torque

마모 정보

- Tool wear (누적 시간)

설비 정보

- Type (L/M/H) one-hot

정비 이력

- time_since_maint

총 **약 9차원**의 연속형 및 이진 피처로 구성된 상태 벡터를 사용합니다.

Action(행동) 공간 설계

이산 행동 공간

복잡한 의사결정을 단순화하여 학습 효율성을 높이기 위해, 두 가지 행동으로 제한했습니다.

a = 0: Continue

정비 없이 생산 계속

a = 1: PM

예방 정비 수행

행동별 영향

Continue 선택 시

- 생산 1건 완료 → **+1 보상**
- Tool wear 증가
- time_since_maint 증가
- 고장 위험 증가

PM 선택 시

- 생산 중단 → **-5 보상**
- tool_wear 리셋
- time_since_maint 리셋
- 고장 위험 감소

Reward(보상) 설계 – 비용 관점

보상 함수는 실제 제조 현장의 경제적 관점을 반영하여 설계되었습니다. 생산 이익, 정비 비용, 고장 비용의 균형을 통해 최적 정책을 학습합니다.

+1	-5	-50
생산 보상	**정비 비용**	**고장 패널티**
정상 생산 시 획득	예방 정비 수행 시	설비 고장 발생 시

보상 함수 구조

> **r_t = 생산 이익 - 정비 비용 - 고장 비용**

Continue & 고장 없음

r_t = +1 (순수 생산 이익)

PM 수행

r_t = -5 (계획된 정비 비용)

고장 발생

추가 -50 (비계획 정지, 수리, 불량 비용)

설계 원칙: C_PM ≪ C_Fail로 설정하여 에이전트가 고장 회피 방향으로 학습하도록 유도

Failure(고장) 판정 규칙 – AI4I 기반

AI4I 2020 데이터셋의 물리 기반 고장 규칙을 시뮬레이터에 구현했습니다. 각 타임스텝마다 5가지 고장 모드를 독립적으로 검사합니다.

TWF (Tool Wear Failure)

조건: wear가 200~240분 범위일 때 확률적 발생

HDF (Heat Dissipation Failure)

조건: (Process Temp - Air Temp) < 8.6K AND Speed < 1380 rpm

PWF (Power Failure)

Power = Torque × (Speed × 2π/60)

조건: Power < 3500W 또는 > 9000W

OSF (Overstrain Failure)

조건: Tool Wear × Torque > Threshold (설비 유형별 상이)

RNF (Random Failure)

조건: 각 타임스텝마다 0.1% 확률

Environment 동작 (1 Step)

상태 관측 (Observe)

현재 상태 s_t를 에이전트에게 전달 (온도, 속도, 토크 등)

행동 선택 (Action)

에이전트가 현재 s_t로부터 a_t 선택 (Continue 또는 PM)

물리적 업데이트 (Physics)

선택된 행동에 따라 wear 및 time_since_maint 업데이트, 새로운 공정 조건 샘플링

고장 판정 (Failure)

5가지 고장(TWF/HDF/PWF/OSF/RNF)을 검사하여 고장 여부 판정

결과 반환 (Return)

보상 r_t, 다음 상태 s_{t+1}, done 여부를 에이전트에게 반환

Agent-Environment 상호작용

- **매** Time Step
- Agent
- **행동** a_t (Action)
- Environment
- **관찰** s_t & **보상** r_t

경험 수집: 매 타임스텝마다 (s_t, a_t, r_t, s_{t+1}, done) 튜플 생성

학습 업데이트: 버퍼에서 샘플링하여 DQN 네트워크 업데이트

DQN 구조 – Q(s,a) 근사 신경망

본 연구는 **Deep Q-Network (DQN)**과 **Double DQN**을 결합하여 Q값 과대평가를 방지하고 안정적인 학습을 달성합니다.

신경망 아키텍처

1. **입력층** (Input Layer)
   9개 상태 변수 (Min-Max 정규화)
2. **은닉층** 1 (Hidden Layer)
   64 units, ReLU 활성화 함수
3. **은닉층** 2 (Hidden Layer)
   64 units, ReLU 활성화 함수
4. **출력층** (Output Layer)
   2 Q-values (Continue vs PM)

학습 안정화 기술

ε-greedy 정책

확률 ε로 무작위 탐색, 나머지는 최적 행동 선택 (argmax_a Q(s,a))

Replay Buffer

10만 개의 경험 저장으로 데이터 상관성 제거

Target Network

일정 주기로 Soft update하여 학습 안정화

Replay Buffer & Target Network

Replay Buffer

경험 재생 버퍼는 DQN 학습의 핵심 구성 요소입니다. 과거 경험을 저장하고 무작위로 샘플링하여 학습에 사용합니다.

- **저장 용량**: 최대 100,000개의 경험 튜플
- **튜플 구조**: (s, a, r, s', done)
- **샘플링 방식**: 미니배치 단위 무작위 추출
- **효과**: 연속된 경험 간 상관성을 줄여 학습 안정성 향상

Target Network

타겟 네트워크는 Q-network의 복제본으로, TD 타겟 계산 시 사용됩니다.

- **역할**: 안정적인 학습 타겟 제공
- **업데이트 방식**: 일정 step마다 Q-network 가중치를 타겟 네트워크로 복사 (Soft update)
- **효과**: 타겟의 급격한 변화를 방지하여 학습 발산 억제

두 기술의 조합은 DQN의 학습 안정성과 수렴성을 크게 향상시킵니다.

DQN 학습 절차 요약

강화학습 에이전트는 다음 4단계의 반복적 프로세스를 통해 점진적으로 최적 정책을 학습합니다.

1. 경험 수집 (Experience Collection)

환경과 상호작용하며 에피소드를 반복 실행합니다. 각 step에서 ε-greedy 정책으로 행동을 선택하고, env.step(a)를 호출하여 (s, a, r, s', done) 경험을 획득합니다.

2. 버퍼 저장 (Buffer Storage)

수집된 모든 경험 튜플을 Replay Buffer에 저장합니다. 버퍼가 가득 차면 가장 오래된 경험부터 제거됩니다 (FIFO 방식).

3. 신경망 학습 (Network Training)

일정 step마다 버퍼에서 미니배치를 무작위 샘플링합니다. TD 타겟 $y = r + \gamma \max_{a'} Q_{target}(s', a')$를 계산하고, 손실 함수 $(y - Q(s,a))^2$를 최소화하도록 Q-network를 업데이트합니다.

4. 타겟 동기화 (Target Update)

target_update_interval마다 Target network의 파라미터를 Q-network의 현재 파라미터로 업데이트합니다. 이를 통해 학습 안정성을 확보합니다.

실험 베이스라인 설정

개발된 DQN 에이전트의 성능을 객관적으로 평가하기 위해, 50번의 테스트 에피소드 동안 다음 세 가지 에이전트를 비교 실험했습니다.

1. Random Agent

매 스텝마다 50% 확률로 무작위 행동을 선택합니다.

예상 결과: 잦은 정비 또는 잦은 고장으로 인해 가장 낮은 성능을 보일 것으로 예상됩니다.

2. Heuristic Agent

규칙 기반(Rule-based) 접근: 공구 마모도가 200분을 초과할 때만 정비를 수행합니다.

예상 결과: 산업 현장의 표준 운영 방식을 대변하는 강력한 비교군으로 작용할 것입니다.

3. DQN Agent

학습된 신경망 정책으로 행동을 결정합니다. 온도, 토크 등 복합 변수를 고려합니다.

목표: 단순 규칙을 넘어, 다변량 센서 데이터를 활용한 최적화된 정비 전략을 학습합니다.

Baseline 정책 vs DQN 정책 비교 계획

Baseline 정책들

다양한 기준선 정책을 설정하여 DQN 에이전트의 성능을 다각도로 평가합니다.

- **No Maintenance**

  정비 없이 항상 continue 선택

- **N-step 정책**

  time_since_maint ≥ N이면 PM 수행

- **Wear-threshold 정책**

  tool wear ≥ W_th이면 PM 수행

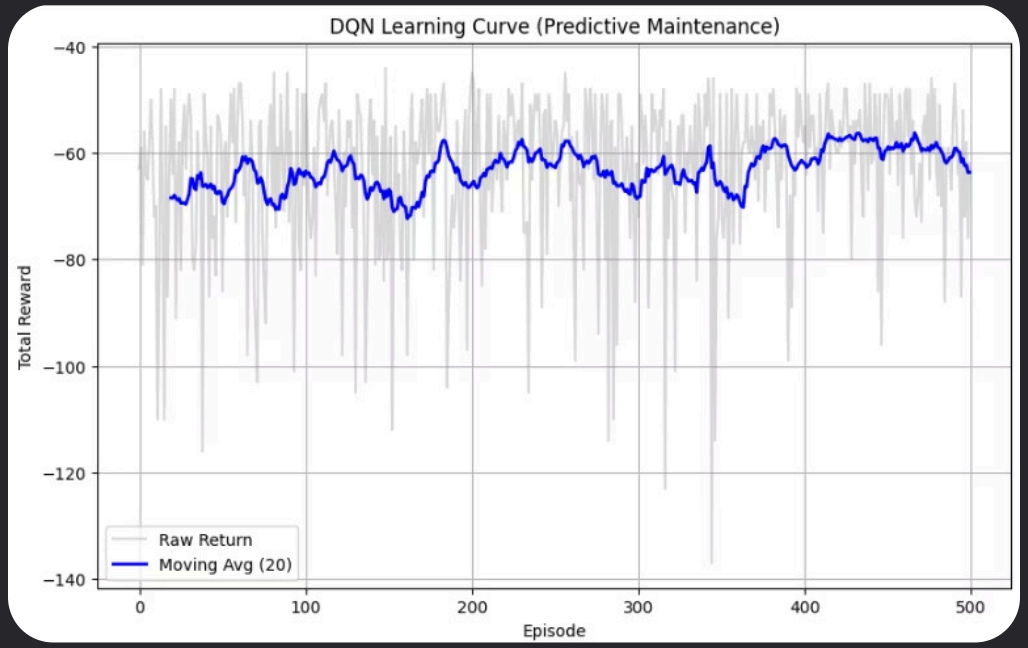
학습된 DQN 정책

DQN 에이전트는 상태 벡터 전체(온도, 속도, 토크, 마모도, 설비 유형, 정비 이력)를 종합적으로 고려하여 장기 비용을 최소화하는 데이터 기반 정책을 학습합니다.

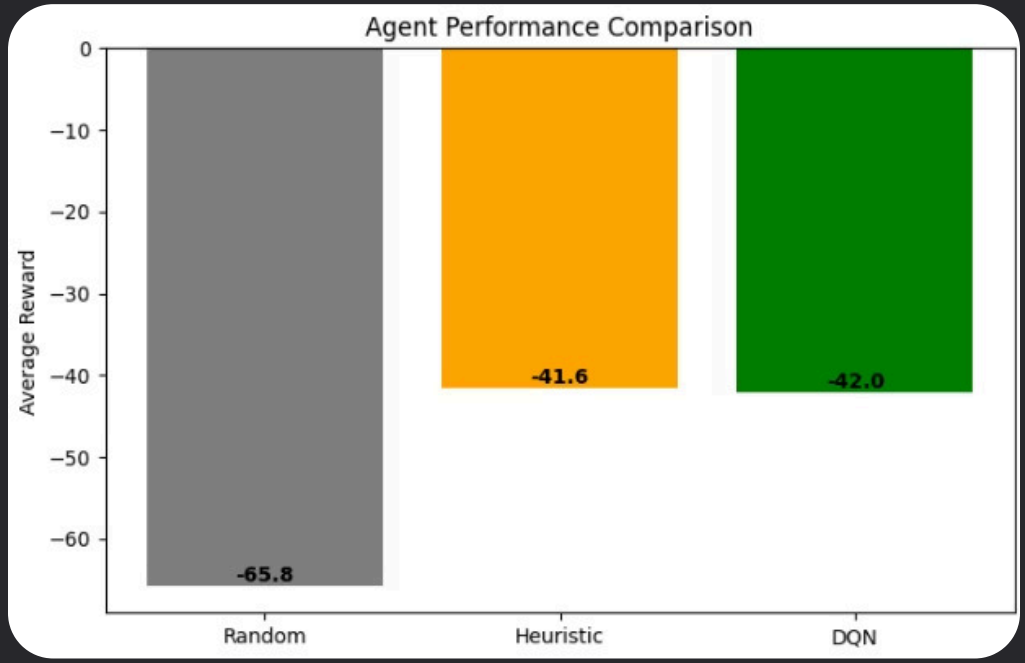
평가 지표

- **평균 Episode Return**: 전체 보상의 평균값
- **Failure Rate**: 에피소드당 고장 발생 비율
- **PM 횟수**: 예방 정비 수행 빈도
- **비용 구성**: 생산 이익, 정비 비용, 고장 비용의 상세 분석

실험 결과 – 그래프 영역



DQN 학습 곡선: 500 에피소드 동안 평균 보상이 점진적으로 상승하며 약 -60 수준으로 수렴. 초기 불안정성 이후 안정적인 학습 패턴을 보임.



에이전트 성능 비교: DQN(-42.0)과 Heuristic(-41.6)이 유사한 성능을 보이며, Random(-65.8) 대비 약 35% 성능 향상. DQN이 효과적인 정비 정책을 학습했음을 입증.

결과 분석 및 고찰

50번의 테스트 에피소드를 통해 DQN 에이전트의 성능을 Heuristic 및 Random 에이전트와 비교 분석했습니다.

성능 동등성 (Parity)

DQN 에이전트의 평균 비용(-42.0)은 Heuristic(-41.6)과 거의 차이가 없습니다. 이는 에이전트가 Random 수준(-67.2)에서 벗어나 **유의미한 정책(마모도 임계값 패턴)**을 성공적으로 학습했음을 의미합니다.

높은 고장률 원인

두 에이전트 모두 모든 에피소드에서 고장이 발생했습니다(50/50). 이는 현재 환경 설정상, 예방 정비만으로는 완벽하게 피할 수 없는 **돌발 고장(RNF)**이나 **급격한 과부하**가 존재함을 시사합니다.

핵심 요약 및 개선 방향

긍정적 측면

- **수렴성 확인**: 무작위 행동 → 전문가 규칙 수준 학습
- 복잡한 물리 기반 고장 모드를 MDP 환경에 성공적으로 반영

개선 필요 사항

- **Local Optima**: 단기 이득 추구 경향 (생산 보상 +1 우선)
- **Reward Shaping**: 고장 패널티를 -100 이상으로 강화하여 리스크 회피 성향 증대

결론 및 향후 과제

결론 요약

1

시뮬레이터 구축 성공

AI4I 2020 데이터의 고장 규칙을 활용해 가상 설비 시뮬레이터를 구축했습니다.

2

복잡한 고장 모드 반영

과열(HDF), 과부하(OSF) 등 물리적 고장 모드를 MDP 환경에 성공적으로 통합했습니다.

3

학습 능력 검증

DQN 에이전트가 전문가 규칙(Heuristic) 수준의 정책을 스스로 학습할 수 있음을 실험적으로 검증했습니다.

향후 연구 방향

1

알고리즘 고도화

PPO, SAC 등 최신 강화학습 알고리즘 적용 및 성능 비교 연구

2

Reward Shaping

고장 패널티 조정 및 안전 마진(safety margin) 보상 추가를 통한 정책 개선

3

Offline RL 적용

실제 현장 데이터(로그)만으로 학습하는 Offline RL 기법 연구 및 산업 현장 적용 가능성 탐색