Tối Ưu Hóa Năng Lượng Microgrid Sử Dụng Deep Reinforcement Learning

Hướng dẫn chi tiết: Yêu cầu đề bài & Cách giải từng phần

DQN PPO PYTHON + PYTORCH

TỔNG QUAN

📋 Cấu Trúc Đề Bài — 6 Phần

📝

Phần 1: Mô Tả Vấn Đề 15%

Giải thích hệ thống microgrid, tại sao là bài toán sequential decision, hạn chế phương pháp truyền thống

🧮

Phần 2: Mô Hình MDP 20%

States (8D), Actions (5), Reward function, Transition dynamics, Episode termination

🤖

Phần 3: Thuật Toán RL 25%

Chọn DQN hoặc Policy Gradient, code Python có comment, giải thích kiến trúc

📈

Phần 4: Phân Tích AI 15%

Agent tối ưu như thế nào, trade-offs, convergence analysis

📊

Phần 5: Kết Quả 15%

Đồ thị, bảng biểu, so sánh với baseline, phân tích kết quả

⚖️

Phần 6: Đạo Đức & Tương Lai 10%

Vấn đề đạo đức, thách thức triển khai, hướng phát triển

💡 **Lưu ý:** Bài nộp là **Google Colab Notebook (.ipynb)** — code + giải thích + đồ thị trong cùng 1 file. Sử dụng PyTorch hoặc TensorFlow.

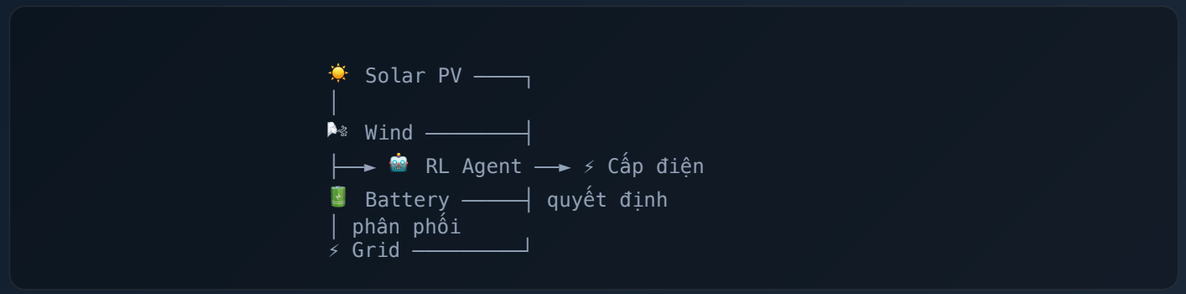
STEP 1

YÊU CẦU ĐỀ BÀI

Phân tích chi tiết từng phần của đề bài — bạn cần làm gì và chuẩn bị gì

PHẦN 1 — 15%

Bài Toán Microgrid



Mục tiêu 3 trong 1:

- ✅ Giảm chi phí điện lưới
- ✅ Tối đa năng lượng tái tạo
- ✅ Đảm bảo đủ điện (reliability)

4 Thành Phần Hệ Thống

Thành phần	Đặc điểm
Solar PV	Cao nhất 10-14h, = 0 ban đêm, có nhiễu
Wind	Ngẫu nhiên, cả ngày lẫn đêm
Battery	100 kWh, hiệu suất 95%
Grid	Giá biến động: thấp off-peak, cao 17-21h

Yêu Cầu Cần Viết

- Giải thích microgrid là gì
- Tại sao là bài toán sequential decision
- Hạn chế phương pháp truyền thống
- Tại sao RL phù hợp hơn

GÓC NHÌN NON-IT

🏠 Câu Chuyện: Người Quản Gia Năng Lượng

Vai trò của AI:
Không phải là kỹ sư điện khô khan, mà là một **Quản gia tận tụy**.
Mục tiêu: Giữ cho chủ nhà luôn vui vẻ (đủ điện) và ví tiền luôn đầy (tiết kiệm).

Hành trình 1 ngày làm việc

🌄 **Sáng (6h-10h):**
Nắng yếu. Dùng tạm pin dư từ đêm. Hạn chế mua siêu thị.

☀️ **Trưa (10h-14h):**
Nắng to! Rau (Solar) đầy vườn.
➜ **Sạc đầy tủ lạnh (Pin)** để dành cho tối.

🌆 **Tối (17h-21h) - CAO ĐIỂM:**
Siêu thị bán giá cắt cổ! 💸
➜ **Tuyệt đối không mua!** Lấy đồ trong tủ lạnh (Pin) ra dùng.

🌙 **Đêm (22h-5h):**
Siêu thị đại hạ giá.
➜ Đi mua đầy tủ lạnh (Sạc) để phòng hờ mai mưa bão.

KIẾN THỨC NỀN 1

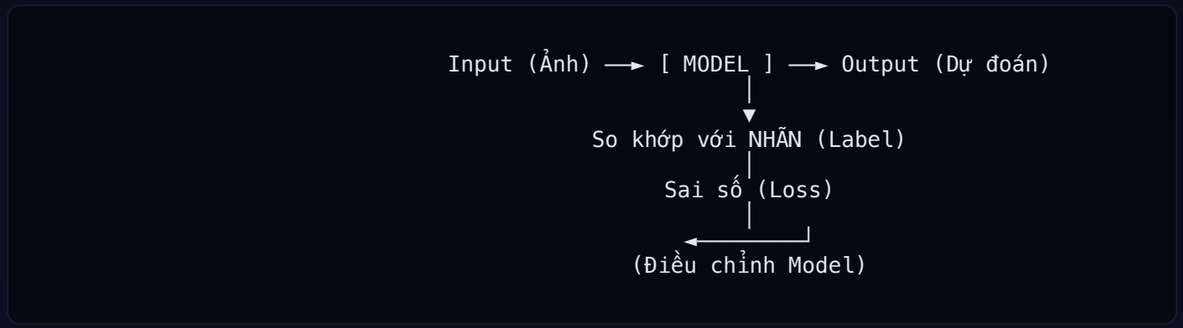
📚 Supervised Learning: "Học Vẹt" Có Đáp Án

Deep Learning truyền thống (CNN, RNN) hoạt động như thế nào?

👨‍🎓 Ví dụ: Luyện thi đại học

Bạn làm bài tập, sau đó mở trang cuối sách xem đáp án.
- Nếu làm sai -> Sửa lại cho đúng đáp án.
- Nếu làm đúng -> Nhớ lấy quy luật.

Đặc điểm cốt lõi:
- Cần bộ dữ liệu có nhãn (Label) chính xác.
- Input: Ảnh con mèo -> Output: Chữ "Mèo".
- Target: **DỰ ĐOÁN (Prediction)**.



❌ Tại sao không dùng cho Microgrid?
Vì không ai biết trước hành động nào là "tốt nhất" ngay lập tức để gán nhãn. "Xả pin bây giờ" là đúng hay sai? -> Cuối ngày mới biết!

KIẾN THỨC NỀN 2

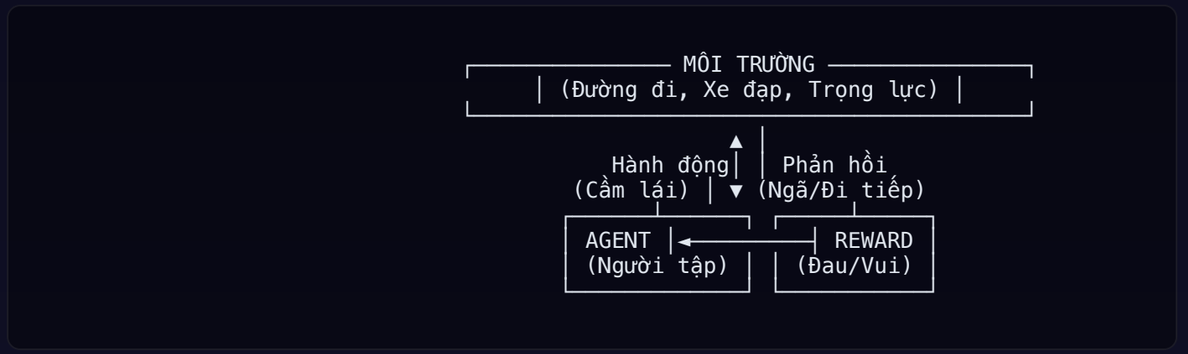
🚴 Reinforcement Learning: "Tập Xe Đạp"

Học qua Thử & Sai (Trial & Error) - Không có đáp án trước

🤕 Ví dụ: Tập đi xe đạp

Không ai dạy bạn "nghiêng người 5.2 độ". Bạn phải tự thử:
- Nghiêng trái quá ➜ Ngã (Đau/Phạt).
- Giữ thăng bằng ➜ Đi được (Vui/Thưởng).
➜ Tự rút ra kinh nghiệm sau nhiều lần ngã.

Đặc điểm cốt lõi:
- Không có nhãn, chỉ có **Thưởng/Phạt (Reward)**.
- Input: Trạng thái -> Output: Hành động.
- Target: **RA QUYẾT ĐỊNH (Decision Making)**.



SO SÁNH

⚔️ Đại Chiến: Supervised DL vs. RL

Deep Learning (DL)

- ✓ **Dữ liệu:** Tĩnh (Static), có sẵn (ImageNet).
- ✓ **Người thầy:** Đáp án có sẵn (Label).
- ✓ **Mục tiêu:** Nhận diện, Dự đoán.
- ✓ **Ví dụ:** Phân loại ảnh, Dịch máy.
- ✓ **Feedback:** Tức thì (Ngay khi đoán sai).

Reinforcement Learning (RL)

- ✓ **Dữ liệu:** Động (Dynamic), tự tạo ra khi tương tác.
- ✓ **Người thầy:** Trải nghiệm (Thử & Sai).
- ✓ **Mục tiêu:** Điều khiển, Chiến thuật sống còn.
- ✓ **Ví dụ:** Chơi Game (Mario), Robot, Microgrid.
- ✓ **Feedback:** Trễ (Cuối ván mới biết thắng/thua).

Tại sao Microgrid cần RL? ⚡

❌ **DL không làm được:** Dữ liệu giá điện & thời tiết thay đổi liên tục, không có kịch bản "chuẩn" để bắt chước mãi mãi.

✅ **RL làm tốt:** Tự thích nghi với môi trường mới, chấp nhận hy sinh lợi ích nhỏ trước mắt để tối ưu hóa đơn tiền điện cuối tháng (Long-term).

SỰ KẾT HỢP

🤝 Deep RL = Mắt Thần + Bộ Não

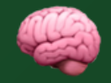
Khi Deep Learning kết hợp với Reinforcement Learning

Deep Learning

Neural Networks (CNN, Dense)

Nhiệm vụ: CẢM NHẬN
Xử lý input phức tạp (State 8 chiều, Lịch sử giá, Thời tiết) thành các đặc trưng (features) hiểu được.

➕



Reinforcement Learning

Q-Learning / Policy Gradient

Nhiệm vụ: RA QUYẾT ĐỊNH
Dựa trên những gì DL "nhìn thấy", chọn hành động tối ưu (Decision Making) để đạt điểm cao nhất.

🚀 **Deep Reinforcement Learning (DRL)** = Khả năng nhận thức của DL + Khả năng hành động của RL.
Đây là công nghệ đứng sau **AlphaGo**, **ChatGPT (RLHF)**, và **Self-driving Cars**.

PHẦN 2 — 20%

Yêu Cầu: Mô Hình Hóa MDP

State Space — 8 biến

#	Biến	Ý nghĩa
1	battery_level	Mức pin [0,1]
2	demand	Nhu cầu điện
3	solar	Sản lượng solar
4	wind	Sản lượng wind
5	price	Giá điện lưới
6	hour_sin	sin(2π×h/24)
7	hour_cos	cos(2π×h/24)
8	prev_action	Action trước

Action Space — 5 hành động rời rạc

ID	Action	Khi nào
0	Xả pin	Peak price + pin đủ
1	Sạc renewable	Solar/wind dư
2	Mua lưới	Thiếu renewable + pin
3	Renew + Xả pin	Ưu tiên renew, hỗ trợ pin
4	Renew + Lưới	Ưu tiên renew, mua grid

Reward Function

```
R = +1.0×renewable − 2.0×grid×price
                        − 5.0×unmet − 0.1×battery_wear
                        + 0.5×(bonus nếu tránh peak)
```

CƠ CHẾ HỌC (REWARD)

🏆 Bảng Điểm Thi Đua Của AI

AI giống như học sinh, học qua việc được thưởng/phạt điểm số

Hành động	Điểm số	Lời thầy cô phê (Ý nghĩa)
🌞 Dùng điện mặt trời	+1.0 điểm	"Giỏi! Biết tận dụng đồ có sẵn, vừa sạch vừa miễn phí."
🔋 Xả pin đúng lúc	+0.5 điểm	"Thông minh! Biết lấy đồ dự trữ ra dùng lúc đắt đỏ."
💸 Mua điện giờ cao điểm	-2.0 điểm	"Hoang phí quá! Sao không dùng pin mà lại đi mua lúc đắt?"
🌑 Để nhà mất điện	-5.0 điểm	"Kỷ luật! Việc này không thể chấp nhận được!" 😡
🏚️ Xả sạc liên tục	-0.1 điểm	"Cẩn thận, làm thế nhanh hỏng pin (Hao mòn)."

🎯 **Mục tiêu:** AI sẽ nghịch ngợm thử đủ cách, nhưng cuối cùng sẽ chọn cách ứng xử nào mang lại **tổng điểm cao nhất** trong ngày.

PHẦN 3 — 25%

🤖 Yêu Cầu: Thuật Toán & Code

Chọn 1 trong 2 nhóm

🔵 Option A: DQN (Value-based)

- Q-Network xấp xỉ Q(s,a)
- Experience Replay Buffer
- Target Network
- Epsilon-greedy exploration

→ Phù hợp: discrete actions, đơn giản

🟣 Option B: PPO (Policy Gradient)

- Actor-Critic Network
- GAE Advantage Estimation
- Clipped Surrogate Objective
- Entropy bonus exploration

→ Phù hợp: policy smooth, scalable

Sản Phẩm Cần Nộp

- ✓ Lựa chọn thuật toán + lý giải tại sao
- ✓ Code Python (PyTorch) — có comment đầy đủ
- ✓ Giải thích kiến trúc neural network
- ✓ Bảng hyperparameters + lý do chọn
- ✓ Pseudocode training loop

⚠️ **QUAN TRỌNG:** Phần này chiếm **25%** — nặng nhất! Code phải chạy được, có comment giải thích, và reproducible (set seed).

QUY TRÌNH HOẠT ĐỘNG

🧠 AI Suy Nghĩ Như Thế Nào? (Decision Loop)

👀

BƯỚC 1: QUAN SÁT
(State Input)

- Pin còn bao nhiêu?
- Trời nắng hay mưa?
- Giá điện đang đắt/rẻ?
- Giờ này là mấy giờ?

➜

🧠

BƯỚC 2: SUY NGHĨ
(DQN / PPO Brain)

Tra cứu kinh nghiệm:
"Lần trước gặp cảnh này mình làm gì?"

Tính toán:
"Nếu Xả pin -> Lời 5đ"
"Nếu Mua điện -> Lỗ 2đ"

⬇️ Ra quyết định

🎬 **BƯỚC 3: HÀNH ĐỘNG (Action)**
Gạt cầu dao: **XẢ PIN** (Discharge)

➜

KẾT QUẢ (Reward)
Nhà đủ điện, Tiết kiệm tiền
(+1.5 điểm)

PHẦN 4, 5, 6

📊 Yêu Cầu: Kết Quả & Đạo Đức

📈

Phần 4: Phân Tích AI 15%

- Agent tối ưu hóa NHƯ THẾ NÀO
- Trade-offs đã cân bằng
- Convergence analysis
- Hạn chế của approach

→ Cần phân tích sâu, không chỉ mô tả

📊

Phần 5: Kết Quả 15%

- Đồ thị training curves
- Bảng so sánh vs baseline
- Episode analysis 24h
- Action distribution

→ Cần đồ thị đẹp + bảng số liệu

⚖️

Phần 6: Đạo Đức 10%

- Công bằng năng lượng
- Bảo mật dữ liệu
- Rủi ro tự động hóa
- Hướng phát triển tương lai

→ Thảo luận có chiều sâu

✅ **Tóm tắt:** Bài tốt = Code chạy được (25%) + MDP hợp lý (20%) + Phân tích sâu (15%) + Đồ thị đẹp (15%) + Mô tả rõ (15%) + Đạo đức (10%)

STEP 2

HƯỚNG DẪN GIẢI CHI TIẾT

Hướng dẫn từng bước giải bài — từ setup đến nộp bài hoàn chỉnh

HƯỚNG DẪN GIẢI

🚀 Bước 1: Setup & Cá Nhân Hóa

Mở Notebook trên Colab

1. **Upload notebook**
   Mở colab.google.com → Upload → chọn file .ipynb
2. **Bật GPU**
   Runtime → Change runtime type → GPU (T4)
3. **Chạy ô cài đặt**
   pip install gymnasium torch numpy matplotlib

Cá Nhân Hóa (BẮT BUỘC)

```
# ⚠️ Thay đổi giá trị này!
SEED = 42 # Mỗi SV khác nhau
EPISODES = 200 # 100–500
LEARNING_RATE = 3e-4

# Gợi ý theo MSSV:
# SV1: SEED=42, LR=3e-4
# SV2: SEED=123, LR=1e-4
# SV3: SEED=456, LR=5e-4
# SV4: SEED=789, LR=2e-4
# SV5: SEED=999, LR=4e-4
```

⚠️ **Phải thay SEED!** Nếu giống bạn khác = bị đánh giá là copy. Kết quả sẽ khác nhau do seed khác.

📝 Cách Viết Mô Tả Vấn Đề 15%

Cấu trúc bài viết

1. **Giới thiệu Microgrid (1 trang)**
   Định nghĩa, 4 thành phần (solar, wind, battery, grid), mỗi thành phần 1 đoạn
2. **Sequential Decision (1/2 trang)**
   3 đặc điểm: temporal coupling, delayed consequences, short vs long term
3. **Hạn chế phương pháp cũ (1/2 trang)**
   Bảng so sánh: Rule-based, LP, MPC, Heuristic
4. **Tại sao RL? (1/2 trang)**
   4 ưu điểm: model-free, adaptive, long-term, handle uncertainty

💡 Tips Viết Điểm Cao

✅ **NÊN:**
- Viết đoạn văn, không chỉ liệt kê bullet
- Cho ví dụ cụ thể (giá điện 17h vs 2h)
- Vẽ sơ đồ hệ thống microgrid
- Trích dẫn paper (Mnih 2015, Schulman 2017)

❌ **TRÁNH:**
- Copy nguyên văn từ nguồn
- Chỉ liệt kê không giải thích
- Bỏ qua phần "tại sao RL"
- Không có ví dụ minh họa

💡 **Góc nhìn Non-IT:** Hãy ví von hệ thống như **"Quản lý bếp ăn gia đình"**:
- **Solar/Wind:** Rau nhà trồng (lúc có lúc không, miễn phí).
- **Grid:** Siêu thị (lúc nào cũng có, giá đổi theo giờ).
- **Battery:** Tủ lạnh (cất rau dùng dần).
- **AI:** Người quản gia tính toán "trưa nay ăn rau vườn hay rau siêu thị" để tiết kiệm nhất.

🧱 Cách Viết MDP 20%

Cần trình bày gì?

1. **Sơ đồ MDP tổng quan**
   Vẽ diagram: Environment → State → Agent → Action → Reward → loop
2. **Bảng State Space (8 biến)**
   Mỗi biến: tên, range, ý nghĩa, LÝ DO đưa vào
3. **Bảng Action Space (5 actions)**
   Mỗi action: mô tả, khi nào nên dùng
4. **Reward Function + Justification**
   Công thức, hệ số, LÝ DO chọn hệ số đó
5. **Transitions & Termination**
   Công thức cập nhật pin, demand, kết thúc episode

💡 **Góc nhìn Non-IT (Reward):** Giống dạy cún cưng! 🐶
• **+1 (Renewable):** Cún làm đúng -> Thưởng bánh.
• **-5 (Mất điện):** Cún cắn đồ -> Mắng phạt.
AI sẽ tự học cách làm đúng để được thưởng nhiều bánh nhất.

💡 Key: Giải thích sin/cos encoding

```
Vấn đề: hour = 0,1,...,23
                        → 23h và 0h "xa nhau" (dist=23)
                        → Nhưng thực tế chỉ cách 1h!

                        Giải pháp:
                        hour_sin = sin(2π × hour / 24)
                        hour_cos = cos(2π × hour / 24)

                        → 23h: (sin≈0, cos≈1)
                        0h: (sin=0, cos=1)
                        → Bây giờ "gần nhau"! ✅
```

🎯 **Điểm mấu chốt:** Phần 2 chiếm 20% — giám khảo muốn thấy bạn **HIỂU** từng thành phần MDP, không chỉ copy công thức. Viết LÝ DO cho mỗi lựa chọn thiết kế.

🔵 Cách Giải: DQN 25%

Kiến Trúc Cần Giải Thích

```
State(8) → [256→ReLU→256→ReLU→128→ReLU] → Q(5)

                        3 thành phần chính:

                        1. Q-Network + Target Network
                        → Target cố định mỗi 1000 steps
                        → Tránh "moving target"

                        2. Experience Replay Buffer (100K)
                        → Random sample batch 64
                        → Phá vỡ temporal correlation

                        3. Epsilon-Greedy (1.0 → 0.01)
                        → Ban đầu explore, sau exploit
```

Công Thức Update

```
target = r + γ × max Q_target(s', a')
                      loss = (Q(s,a) - target)²
                      θ ← θ - α × ∇loss (Adam, lr=1e-4)
```

💡 **Góc nhìn Non-IT (DQN):**
DQN giống như **học vẹt bằng Flashcards**.
• **Experience Replay:** Xáo trộn thẻ bài để học ngẫu nhiên, tránh học tủ.
• **Target Network:** Giữ nguyên đáp án một lúc để ôn tập cho nhớ, tránh đề bài thay đổi liên xoành xoạch.

Trong Bài Viết, Cần Có:

- ✓ Tại sao chọn DQN (so sánh thuật toán)
- ✓ Sơ đồ kiến trúc neural network
- ✓ Giải thích Experience Replay
- ✓ Giải thích Target Network
- ✓ Giải thích Epsilon-Greedy
- ✓ Bảng hyperparameters + lý do
- ✓ Pseudocode training loop
- ✓ Code snippet có comment

📁 **File sẵn:**
`Microgrid_DQN_Simple.ipynb` — Chạy 3 bước
`REPORT_DQN.md` — Báo cáo chi tiết
→ Dùng làm tham khảo, **phải viết lại bằng lời mình**

🟣 Cách Giải: PPO 25%

Kiến Trúc Actor-Critic

```
State(8) → Shared[128→Tanh→128→Tanh]
                        |
                   ┌────┴────┐
                   Actor Critic
                   128→5 128→1
                   Softmax (linear)
                   → π(a|s) → V(s)
```

Công Thức PPO

```
GAE: A_t = Σ(γλ)^l × δ_{t+l}
                        δ_t = r + γV(s') − V(s)

                        Clip: L = min(ratio×A,
                        clip(ratio,0.8,1.2)×A)

                        Total: L_π + 0.5×L_v − 0.01×H
```

> 💡 **Góc nhìn Non-IT (PPO):**
> PPO giống như **Huấn luyện viên thể thao** 🏃 .
> • Không bắt thay đổi toàn bộ kỹ thuật ngay (dễ chấn thương).
> • Chỉ chỉnh sửa **từng chút một (Clipped)** để tiến bộ vững chắc.

DQN vs PPO — Chọn cái nào?

Tiêu chí	DQN	PPO
Độ khó code	⭐⭐	⭐⭐⭐
Ổn định	Tốt	Rất tốt
Điểm ấn tượng	Trung bình	Cao
Dễ giải thích	Dễ	Khó hơn

> 💡 **Khuyến nghị:**
> • Nếu muốn **an toàn** → DQN (đơn giản, dễ giải thích)
> • Nếu muốn **điểm cao** → PPO (ấn tượng hơn, policy gradient)
> • Nếu muốn **nổi bật** → Làm CẢ HAI và so sánh!

KIẾN THỨC CHUYÊN SÂU

🔵 DQN Deep Dive: Học Qua Ký Ức (Replay)

Tại sao DQN cần 2 mạng thần kinh và 1 bộ nhớ?

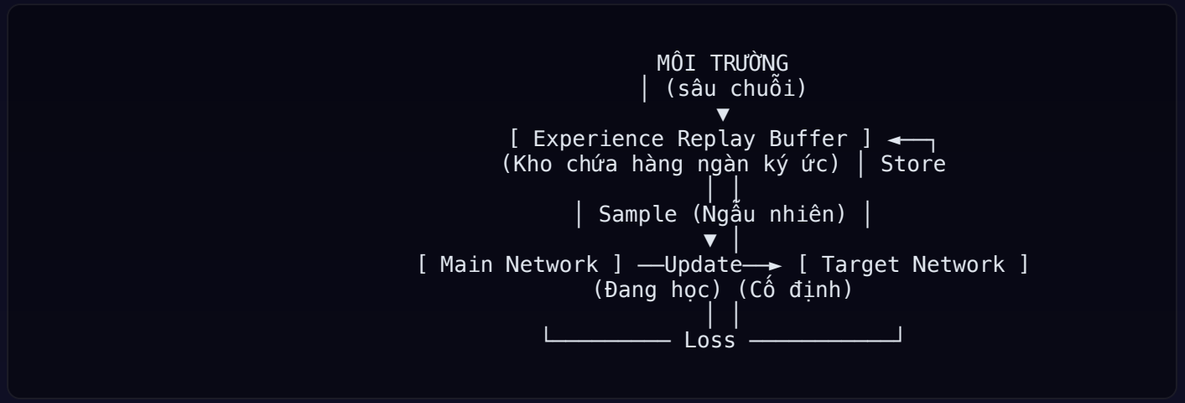
1. Experience Replay (Bộ Nhớ Hồi Tưởng)

Vấn đề: Học trực tiếp từ sự kiện vừa xảy ra giống như "học vẹt", dễ quên bài cũ.
Giải pháp: Lưu mọi trải nghiệm (S, A, R, S') vào một cái kho (Buffer). Khi học, bốc ngẫu nhiên 1 nắm từ kho ra.
➜ *Giống như ôn thi bằng thẻ Flashcards trộn lẫn lộn.*

2. Target Network (Mạng Mục Tiêu)

Vấn đề: Vừa học vừa sửa đáp án (Moving Target) rất khó hội tụ.
Giải pháp: Copy 1 bản sao của não bộ (Target Network) và giữ nguyên nó trong 1000 bước. Chỉ dùng nó để chấm điểm, không cho nó học.
➜ *Giống như giữ nguyên đề thi trong 1 tuần để học sinh ôn, không đổi đề liên tục.*

Sơ Chế Hoạt Động DQN



Off-Policy: DQN có thể học từ ký ức của "phiên bản cũ" của chính nó. Không cần phải là dữ liệu mới nhất.

KIẾN THỨC CHUYÊN SÂU

🟣 PPO Deep Dive: Actor-Critic & Clipping

Tại sao PPO ổn định và được dùng cho ChatGPT?

1. Actor-Critic (Diễn viên & Phê bình)

AI được chia làm 2 nhân cách:

- **Actor (Diễn viên):** Quyết định hành động (Đi đâu? Làm gì?).
- **Critic (Nhà phê bình):** Đánh giá hành động đó hay hay dở (Cho điểm V(s)).

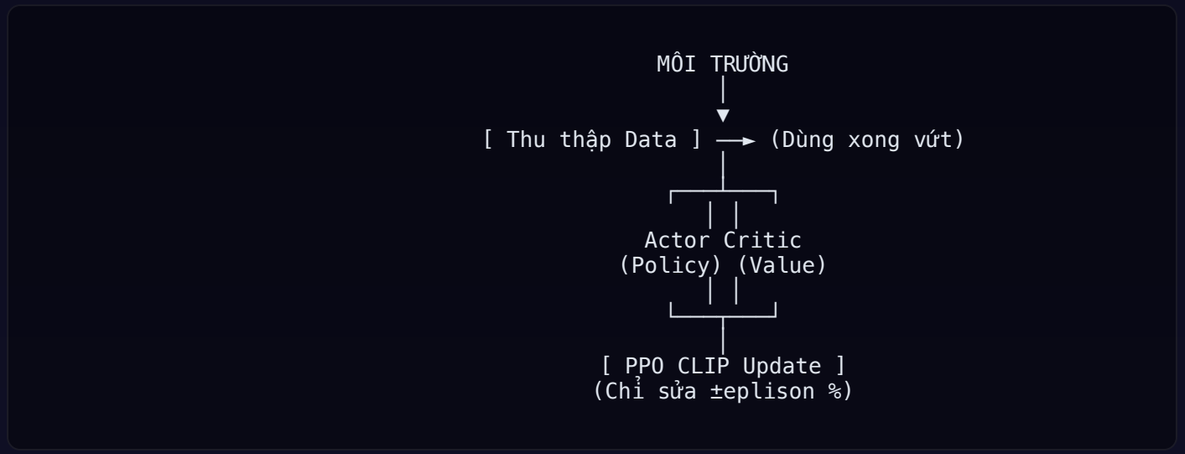
2. Clipped Objective (Cắt Tỉa)

Vấn đề: Nếu sửa đổi não bộ quá nhiều một lúc, AI sẽ bị "sốc" và quên hết những gì đã học.

Giải pháp: Chỉ cho phép update trong khoảng nhỏ (ví dụ ±20%). Nếu vượt quá -> Cắt bỏ (Clip).

➜ *Giống HLV chỉnh sửa kỹ thuật: "Thấp tay xuống một chút thôi", không bắt đổi thế đánh hoàn toàn.*

Sơ Chế Hoạt Động PPO



On-Policy: PPO chỉ học từ dữ liệu nóng hổi vừa tạo ra. Học xong là vứt bỏ dữ liệu, không lưu trữ lại.

TỔNG KẾT

⚔️ Đại Chiến Thuật Toán: DQN vs. PPO

Khi nào nên dùng cái nào cho bài toán Microgrid?

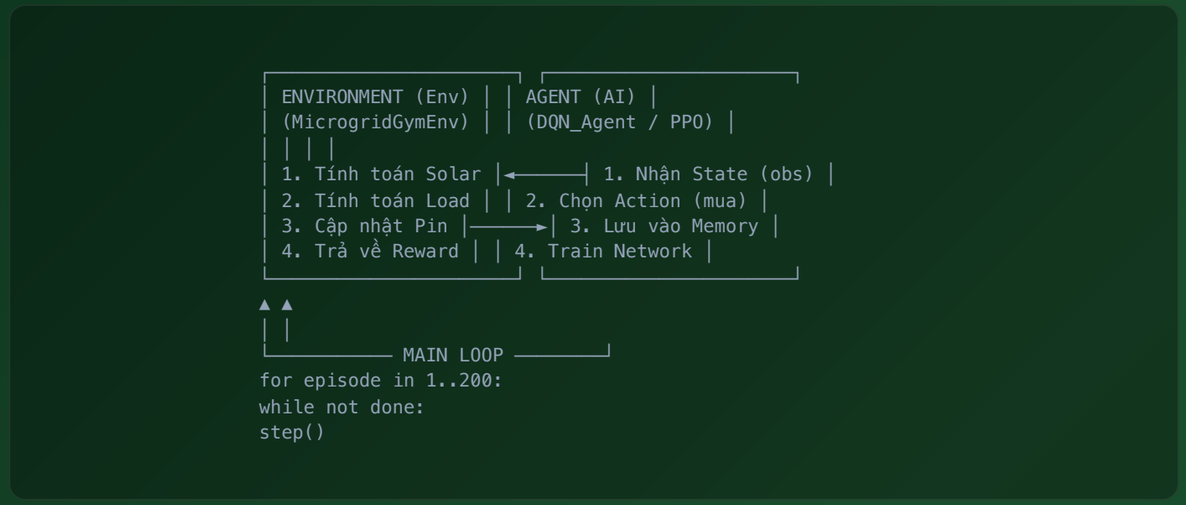
Tiêu chí	🔵 DQN (Deep Q-Network)	🟣 PPO (Proximal Policy Opt)
Loại thuật toán	**Value-based:** Học bảng giá trị Q(s,a) để chọn hành động có giá cao nhất.	**Policy-based:** Học trực tiếp chiến thuật (Policy) và hàm giá trị (Value).
Cách học data	**Off-Policy:** Học lại ký ức cũ (Replay Buffer). ➜ Tiết kiệm data (Sample Efficient).	**On-Policy:** Chỉ học data mới nhất. ➜ Tốn data hơn, nhưng ổn định hơn.
Không gian hành động	Chỉ hỗ trợ **Rời rạc (Discrete)**. (Bật/Tắt, 0/1/2).	Hỗ trợ cả **Rời rạc & Liên tục**. (Điều chỉnh van 0.5%, 1.2%...).
Độ ổn định	Thấp hơn. Dễ bị dao động nếu hyperparams không chuẩn.	**Rất cao** nhờ cơ chế Clipping. Dễ tune hơn.
Kết luận	✅ Tốt cho người mới bắt đầu. ✅ Tốt cho bài toán đơn giản, ít actions.	✅ Chuẩn mực hiện đại (SOTA). ✅ Dùng cho bài toán phức tạp, robot.

> 🏆 **Khuyến nghị cho Đồ Án:** Hãy bắt đầu với **DQN** vì nó dễ hiểu và dễ debug hơn. Sau đó nâng cấp lên **PPO** để lấy điểm cộng và so sánh hiệu suất!

IMPLEMENTATION DETAILS

💻 Cấu Trúc Code & Dòng Chảy Dữ Liệu

Sơ Đồ Hệ Thống (Block Diagram)



Lớp (Class) quan trọng:
- `MicrogridGymEnv`: Mô phỏng vật lý (pin, giá, thời tiết).
- `DQNAgent` / `PPOAgent`: Bộ não chứa Neural Network.
- `ReplayBuffer`: Bộ nhớ (chỉ dùng cho DQN).

Snippet: Vòng lặp huấn luyện

```
for episode in range(EPISODES):
state = env.reset()
done = False
total_reward = 0

while not done:
# 1. AI chọn hành động
action = agent.act(state)

# 2. Môi trường phản hồi
next_state, reward, done, _ = env.step(action)

# 3. Lưu kinh nghiệm (DQN)
agent.remember(state, action, reward, next_state, done)

# 4. Học (Training)
agent.replay(BATCH_SIZE)

state = next_state
total_reward += reward

print(f"Episode {episode}: Reward = {total_reward}")
```

HƯỚNG DẪN GIẢI — PHẦN 4 & 5

📊 Cách Viết Phân Tích & Kết Quả

Phần 4: Phân Tích AI (15%)

1. **Bảng hành vi agent theo giờ**
   0-6h: Grid+Renew, 10-14h: Sạc pin, 18-21h: Xả pin. Giải thích TẠI SAO
2. **3 Trade-offs chính**
   Cost vs Renewable, Battery wear vs Value, Short vs Long term
3. **Hạn chế & cải tiến**
   Discrete actions, no forecasting, sim-to-real gap

Phần 5: Kết Quả (15%)

1. **Bảng metrics chính**
   Agent vs Random: Reward, Cost, Renewable%, Unmet%
2. **3 đồ thị bắt buộc**
   Training curves, Agent vs Random bars, Episode 24h analysis
3. **Thảo luận kết quả**
   Giải thích CON SỐ, không chỉ liệt kê. "Tại sao reward tăng?"

📊 Notebook đã tự sinh đồ thị — chỉ cần **chạy Bước 3** rồi **phân tích kết quả**.

💡 **Góc nhìn Non-IT (Chiến thuật):**
AI học được chiêu **"Buôn gian bán lận"**:
• **Đêm/Sáng:** Giá rẻ -> Tích trữ hàng (Sạc pin).
• **Trưa:** Hàng free (Solar) -> Gom hết về kho.
• **Chiều tối:** Giá đắt -> Xả kho bán kiếm lời (Xả pin).

⚖️ Cách Viết Đạo Đức & Tương Lai 10%

4 Vấn Đề Đạo Đức

1. Công bằng năng lượng
AI ưu tiên tối ưu cost → có thể bỏ qua nhóm yếu thế. Giải pháp: thêm constraint demand satisfaction tối thiểu.

2. Rủi ro tự động hóa
AI sai → mất điện → thiệt hại. Giải pháp: human-in-the-loop, fallback rule-based.

3. Bảo mật dữ liệu
Data tiêu thụ tiết lộ thói quen sinh hoạt. Giải pháp: differential privacy, GDPR.

4. Trách nhiệm pháp lý
AI gây lỗi → ai chịu trách nhiệm? Developer? Operator? Cần quy định rõ.

5 Hướng Phát Triển Tương Lai

🔗 **Multi-Agent RL**
Nhiều microgrid hợp tác, chia sẻ năng lượng dư

🔮 **Demand Forecasting**
Tích hợp LSTM dự đoán demand 24h → proactive

🎯 **Continuous Actions**
DDPG/SAC cho control chính xác kW

🔄 **Transfer Learning**
Train 1 lần, deploy nhiều microgrid

🛡️ **Safe RL**
Constrained optimization, đảm bảo an toàn

CHECKLIST

✅ Quy Trình Hoàn Thành & Nộp Bài

Checklist Trước Khi Nộp

1. **Thay đổi SEED + hyperparameters**
   Đảm bảo kết quả khác bạn khác
2. **Chạy notebook từ đầu đến cuối**
   Runtime → Run all, đảm bảo không lỗi
3. **Viết phân tích cho mỗi phần**
   Markdown cells giải thích + phân tích kết quả
4. **Kiểm tra đồ thị + bảng biểu**
   Có title, legend, axis labels đầy đủ
5. **Thêm references**
   APA/IEEE format, ít nhất 4-5 papers
6. **Download và nộp**
   .ipynb (notebook) + .pdf (report nếu cần)

📁 Files Tham Khảo Có Sẵn

File	Dùng để
Microgrid_DQN_Simple.ipynb	Chạy DQN 3 bước
Microgrid_PPO_Simple.ipynb	Chạy PPO 3 bước
ESSAY_TEMPLATE.md	Template bài essay
REPORT_DQN.md	Tham khảo giải thích DQN
REPORT_PPO.md	Tham khảo giải thích PPO
REPORT.md	Báo cáo tổng hợp mẫu

> 🏆 **Mẹo điểm cao:** Làm CẢ HAI thuật toán (DQN + PPO) và so sánh → ấn tượng giám khảo!

🎓

Chúc Các Bạn
Hoàn Thành Tốt Đồ Án!

Bắt đầu sớm • Thay đổi SEED • Viết phân tích sâu • Trích dẫn đầy đủ

📁 Tài liệu: REPORT_DQN.md | REPORT_PPO.md | ESSAY_TEMPLATE.md

📓 Notebooks: Microgrid_DQN_Simple.ipynb | Microgrid_PPO_Simple.ipynb