|  |  |  |
| --- | --- | --- |
|  | BAN CƠ YẾU CHÍNH PHỦ  “HỌC VIỆN KỸ THUẬT MẬT MÃ” | Mẫu 2 |

BÁO CÁO CHUYÊN ĐỀ SỐ 2.5

“Nghiên cứu một số mô hình học sâu tiêu biểu học liên hợp học phân tán”

NHIỆM VỤ: “Nghiên cứu và ứng dụng nền tảng học sâu để xây dựng hệ thống phát hiện mã độc trực tuyến”.

Mã số: 06/2022/CB.

Cơ quan chủ trì: Học viện Kỹ thuật Mật mã

Chủ nhiệm: ThS. Lê Đức Thuận

Hà Nội - 2022

|  |  |  |
| --- | --- | --- |
|  | BAN CƠ YẾU CHÍNH PHỦ  “HỌC VIỆN KỸ THUẬT MẬT MÃ” |  |

BÁO CÁO CHUYÊN ĐỀ SỐ 2.5

“Nghiên cứu một số mô hình học sâu tiêu biểu học liên hợp học phân tán”

NHIỆM VỤ: “Nghiên cứu và ứng dụng nền tảng học sâu để xây dựng hệ thống phát hiện mã độc trực tuyến”.

Mã số: 06/2022/CB.

Cơ quan chủ trì: Học viện Kỹ thuật Mật mã

Chủ nhiệm: ThS. Lê Đức Thuận

|  |  |
| --- | --- |
| **Người thực hiện chuyên đề** | **Cơ quan chủ trì** |
| *(Họ tên và chữ ký)* | *(Họ tên và chữ ký)* |

Hà Nội - 2022

MỤC LỤC

[MỤC LỤC 3](#_Toc116022457)

[DANH MỤC HÌNH ẢNH 4](#_Toc116022458)

[1. HỌC LIÊN HỢP 5](#_Toc116022459)

[1.1. Tổng quan 5](#_Toc116022460)

[1.2. Các khái niệm 6](#_Toc116022461)

[1.3. Hướng triển khai 7](#_Toc116022462)

[Dựa trên value (Value-based) 7](#_Toc116022463)

[Dựa trên policy (Policy-based) 7](#_Toc116022464)

[Dựa trên model (Model-based) 7](#_Toc116022465)

[1.4. Các thuật toán 8](#_Toc116022466)

[1.4.1. Các thuật toán dựa trên value 8](#_Toc116022467)

[1.4.2. Các thuật toán dựa trên policy 13](#_Toc116022468)

[1.5. Các thư viện 17](#_Toc116022469)

[1.5.1. OpenAI Baselines 17](#_Toc116022470)

[1.5.2. Stable Baselines 18](#_Toc116022471)

[1.5.3. Keras-RL 18](#_Toc116022472)

[1.5.4. TF Agents 18](#_Toc116022473)

[1.5.5. PyQlearning 18](#_Toc116022474)

[1.5.6. Tensorforce 18](#_Toc116022475)

[2. HỌC PHÂN TÁN 19](#_Toc116022476)

[TÀI LIỆU THAM KHẢO 23](#_Toc116022477)

# DANH MỤC HÌNH ẢNH

[Hình 1.1: Ý tưởng học tăng cường 6](#_Toc116022478)

[Hình 1.2: Các hướng triển khai 7](#_Toc116022479)

[Hình 1.3: Mô tả thuật toán MDP 8](#_Toc116022480)

[Hình 1.4: Mô tả Q learning và Deep Q learning 10](#_Toc116022481)

[Hình 1.5: Mô tả xác xuất và giá trị phân phối 10](#_Toc116022482)

[Hình 1.6: Đồ thị biểu diễn các thuật toán QR-DQN 13](#_Toc116022483)

[Hình 1.7: So sánh A3C và A2C 17](#_Toc116022484)

[Hình 2.1: Tổng quan về huấn luyện phân tán mạng học sâu 20](#_Toc116022485)

[Hình 2.2: Mô hình huấn luyện phân tán chia theo lớp 20](#_Toc116022486)

[Hình 2.3: Mô hình huấn luyện phân tán chia theo node 21](file:///D:\DT_Ban%202022\bản%20word\Báo%20cáo%20tiến%20độ\Nội%20dung%202\BC_CĐ%202.5.%20Nghiên%20cứu%20một%20số%20mô%20hình%20học%20sâu%20tiêu%20biểu%20học%20liên%20hợp%20học%20phân%20tán.docx#_Toc116022487)

[Hình 2.4: Phương pháp song song hóa dữ liệu 22](#_Toc116022488)

# 1. HỌC LIÊN HỢP

## 1.1. Tổng quan

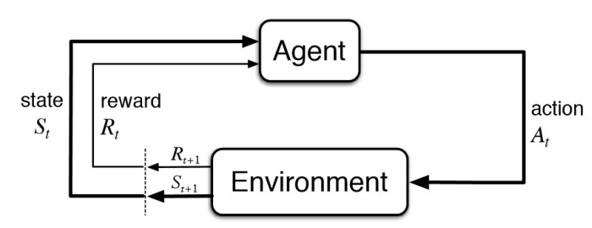
Học liên hợp hay còn gọi là học tăng cường (Reinforcement Learning - RL) là một phương pháp học máy nhằm đạt được mục tiêu nhờ tương tác trực tiếp với môi trường.

* **Đặc điểm**
* Trong RL, agent không được hướng dẫn về môi trường cũng như action nó cần đi
* Dựa trên quá trình truy cập và thử lại.
* Agent thực hiện hành động tiếp theo và thay đổi trạng thái dựa trên phản hồi của action trước đó.
* Agent có thể được nhận reward muộn
* Môi trường là ngẫu nhiên, và agent cần khám phá nó để đạt được reward lớn nhất.
* **Ưu điểm**
* Có thể giải quyết các vấn đề phức tạp và bậc cao hơn. Ngoài ra, các giải pháp thu được sẽ rất chính xác. Do kỹ thuật học mô phỏng theo con người.
* Có thể sử dụng với các mạng Neural Network: Deep Reinforcement Learning.
* Là mô hình học hỏi liên tục nên một sai lầm được thực hiện trước đó sẽ khó có thể xảy ra trong tương lai.
* Có thể xây dựng nhiều mô hình giải quyết vấn đề khác nhau bằng cách sử dụng phương pháp Reinforcement Learning.
* Khi nói đến việc tạo mô phỏng, phát hiện đối tượng trong ô tô tự động, rô bốt, v.v., Reinforcement Learning đóng một vai trò lớn.
* Ngay cả khi không có dữ liệu huấn luyện, nó sẽ học được thông qua kinh nghiệm mà nó có được từ việc xử lý dữ liệu huấn luyện.
* Đối với các vấn đề khác nhau, có vẻ phức tạp đối với con người, RL cung cấp các mô hình hoàn hảo để giải quyết chúng.
* **Nhược điểm**
* Không phù hợp với các vấn đề đơn giản
* Yêu cầu lượng lớn dữ liệu huấn luyện để có kết quả chính xác - tiêu tốn nhiều thời gian và sức mạnh tính toán.
* Yêu cầu mô phỏng hoặc xây dựng mô hình thực tế cho việc huấn luyện.
* **Ứng dụng**

Dưới đây là một số ứng dụng của Reinforcement Learning:

* Định hướng robot, robot đi bộ, …
* Tự động hóa dây chuyền
* Lập chiến lược kinh doanh
* Học máy và xử lý dữ liệu
* Điều khiển giao thông
* Hệ thống gợi ý mua hàng
* Áp dụng trong trò chơi: Cờ, Trò chơi chiến thuật, …
* Xe tự lái

## 1.2. Các khái niệm



Hình 1.1: Ý tưởng học tăng cường

**Agent**: Một tác nhân có thể quan sát và tương tác với môi trường.

**Environment**: Môi trường nơi agent quan sát và tương tác.

**Action**: Hành động của agent bên trong environment.

**State**: Trạng thái của environment sau mỗi hành động của agent.

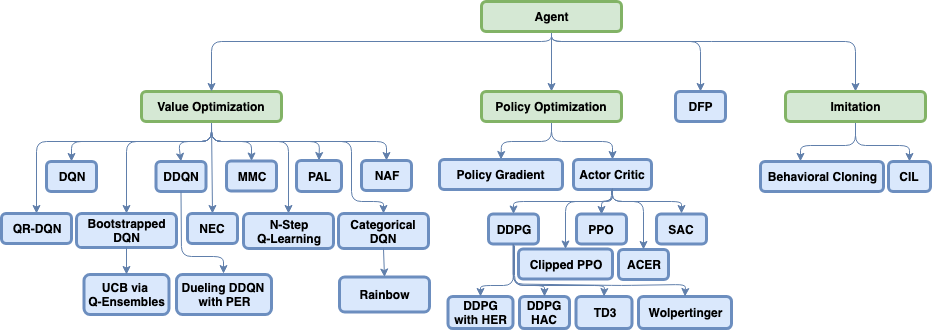
**Reward**: Một phản hồi từ environment đến agent để đánh giá action. Tín hiệu này chỉ ra hướng đi tốt tiếp theo.

**Policy**: Chiến thuật được agent sử dụng cho action tiếp theo dựa trên state.

**Value**: Ngược lại với reward, value chỉ ra hướng đi tốt về dài hạn. Có thể nói, value là tổng reward mà agent tích lũy trong tương lai từ một state.

**Q-value**: Giống với value nhưng nhận thêm một tham số khác là action hiện tại.

## 1.3. Hướng triển khai



Hình 1.2: Các hướng triển khai

### Dựa trên value (Value-based)

Hướng tiếp cận dựa trên model là việc tìm một hàm value tối ưu: Tối đa hóa value từ state trong mọi policy.

### Dựa trên policy (Policy-based)

Hướng tiếp cận dựa trên Policy tìm policy tối ưu nhằm tối đa hóa reward trong tương lai. Phương hướng này có 2 loại policy chính:

Xác định (Deterministic): Các action giống nhau được tạo ra bởi policy π vào tất cả state

Ngẫu nhiên (Stochastic): Trong policy này, xác suất quyết định action

### Dựa trên model (Model-based)

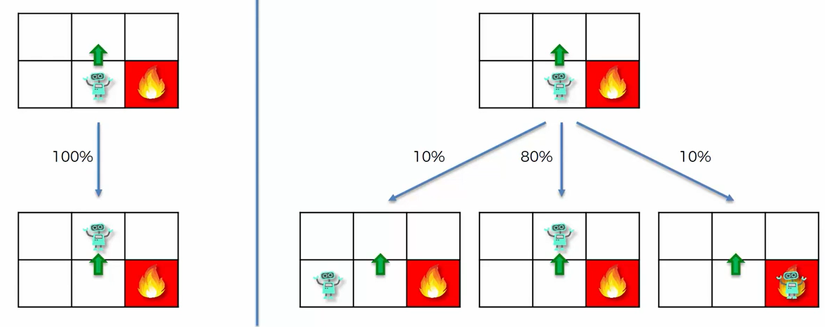
Trong hướng tiếp cận này, một model ảo được tạo ra cho môi trường, agent khám phá môi trường này để học. Không có giải pháp cụ thể hay thuật toán cho hướng triển khai này.

## 1.4. Các thuật toán

### 1.4.1. Các thuật toán dựa trên value

#### 1.4.1.1. Markov Decision Process (MDP)

MDP là một framework giúp agent đưa ra quyết định tại một state nào đó. Để áp dụng được framework này, ta giả sử các states có thuộc tính Markov (Markov Property): mỗi state chỉ phụ thuộc vào state trước nó vào xác suất chuyển đổi giữa 2 states này. Nghe hơi khó hiểu nhưng thực ra rất đơn giản. Đầu tiên, sao lại có “xác suất chuyển đổi giữa 2 states”? Trong một “thế giới hoàn hảo”, nếu ta lặp lại một hành động thì sẽ cho ra 2 kết quả giống hệt nhau. Nhưng mọi việc không đơn giản như vậy vì hầu hết mọi sự việc đều là các tiến trình ngẫu nhiên (stochastic process)



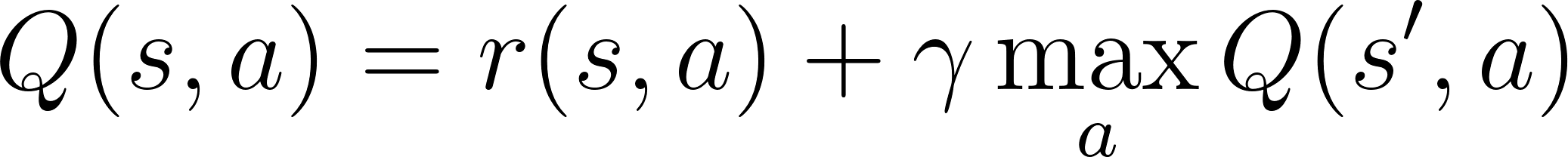
Hình 1.3: Mô tả thuật toán MDP

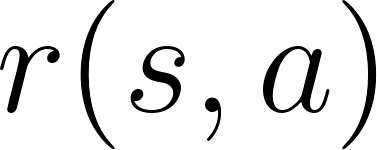
Ở hình 1.3. nếu như agent quyết định đi lên và environment trả lại state “ô trên” với xác suất 100% thì sẽ không có gì để nói. Thay vào đó, kết quả agent nhận được lại là 80% state “ô trên”, 10% state “ô trái” và 10% “ô phải”.

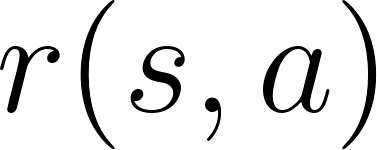
Một thuộc tính của Markov property là tính “không nhớ” (memoryless), tức là tại vị trí hiện tại, xác suất di chuyển của agent luôn không đổi.

#### 1.4.1.2. Q-Learning

Để áp dụng MDP vào RL, ta sử dụng giá trị Q-value để có thể chọn action để đạt reward lớn nhất. Q-value được tính theo công thức:

[](https://www.codecogs.com/eqnedit.php?latex=%20Q(s%2C%20a)%20%3D%20r(s%2C%20a)%20%2B%5Cgamma%20%5Cmax_%7Ba%7D%20Q(s'%2C%20a)#0)

Trong đó [](https://www.codecogs.com/eqnedit.php?latex=Q(s%2Ca)#0) là Q-value khi thực hiện action [](https://www.codecogs.com/eqnedit.php?latex=a#0) tại state [](https://www.codecogs.com/eqnedit.php?latex=s#0); [](https://www.codecogs.com/eqnedit.php?latex=r(s%2Ca)#0) là reward nhận được; [](https://www.codecogs.com/eqnedit.php?latex=s'#0)là state kế tiếp. γ là hệ số discount, đảm bảo càng “xa” đích Q-value càng nhỏ

Công thức này cho thấy Q-value của action [](https://www.codecogs.com/eqnedit.php?latex=a#0) tại state [](https://www.codecogs.com/eqnedit.php?latex=s#0) bằng reward [](https://www.codecogs.com/eqnedit.php?latex=r(s%2Ca)#0) cộng với Q-value lớn nhất của các states [](https://www.codecogs.com/eqnedit.php?latex=s'#0) tiếp theo khi thực hiện các action [](https://www.codecogs.com/eqnedit.php?latex=a#0). Như vậy, chỉ với công thức đơn giản chúng ta có thể tạo ra một ma trận state-action như một lookup table. Từ đó với mỗi state agent chỉ cần tìm action nào có Q-value lớn nhất là xong. Tuy nhiên, RL là một stochastic process nên Q-value ở thời điểm trước và sau khi thực hiện action sẽ khác nhau. Khác biệt này gọi là Temporal Difference:

[](https://www.codecogs.com/eqnedit.php?latex=%20TD(a%2C%20s)%3DR(s%2C%20a)%20%2B%20%5Cgamma%20%5Cmax_%7Ba'%7D%20Q(s'%2C%20a')%20-%20Q_%7Bt-1%7D(s%2C%20a)%20#0)

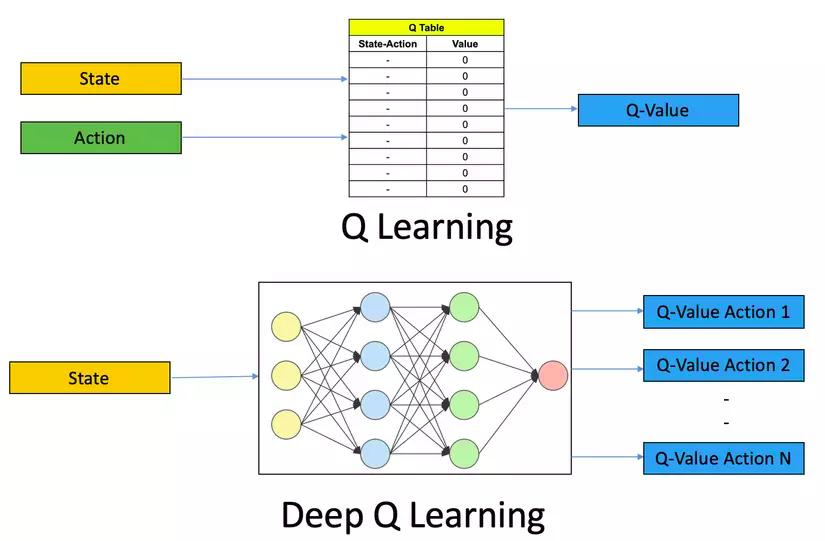
Như vậy, ma trận Q(s, a) cần phải cập nhật trọng số dựa trên TD:

[](https://www.codecogs.com/eqnedit.php?latex=%20Q_t(s%2C%20a)%20%3D%20Q_%7Bt-1%7D(s%2C%20a)%20%2B%20%5Calpha%20TD_t(a%2C%20s)#0)

Trong đó α là learning rate. Qua các lần agent thực hiện actions, [](https://www.codecogs.com/eqnedit.php?latex=Q(s%2Ca)#0) sẽ dần hội tụ. Quá trình này chính là Q-Learning

#### 1.4.1.3. Deep Q-Learning (Deep Q-Network)

Mục đích của chúng ta là chọn ra action thích hợp cho một state nào đó; hay nói cách khác, chúng ta state làm input, output là một action. Vì vậy, có thể thay thế lookup table [](https://www.codecogs.com/eqnedit.php?latex=Q(s%2Ca)#0) bằng một Neural Network

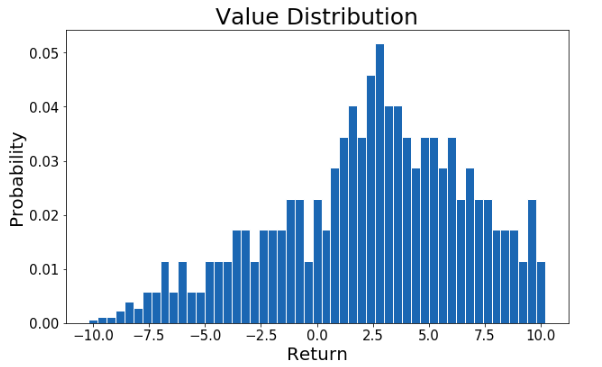


Hình 1.4: Mô tả Q learning và Deep Q learning

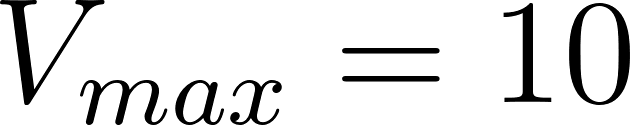
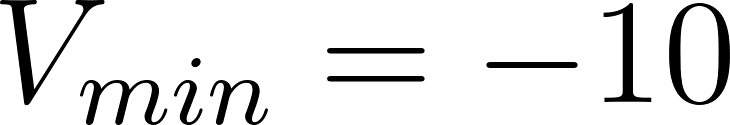
#### 1.4.1.4. C51

Thuật toán C51 đầu tiên tìm trước một dự đoán và sau đó giảm thiểu sự phân tán giữa cập nhật Bellman và dự đoán.

Trong Categorical DQN, các giá trị trả về bị giới hạn trong một tập các giá trị cố định rời rạc (51). Xác suất của từng giá trị được học trong quá trình tương tác với môi trường.

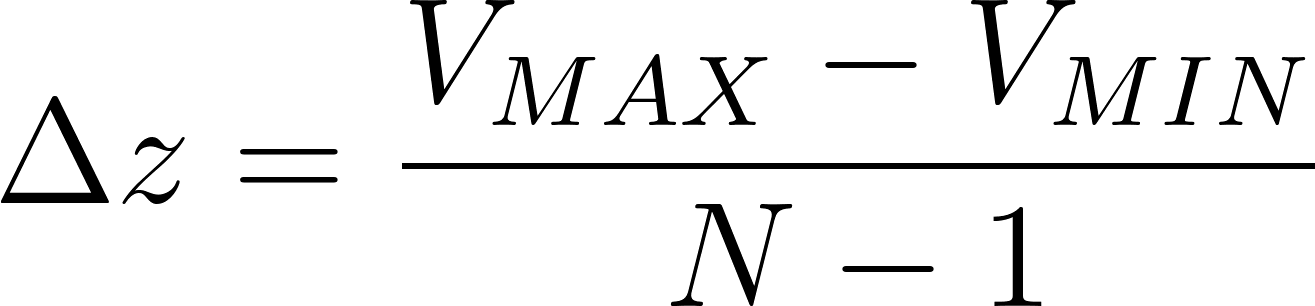


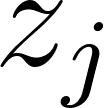
Hình 1.5: Mô tả xác xuất và giá trị phân phối

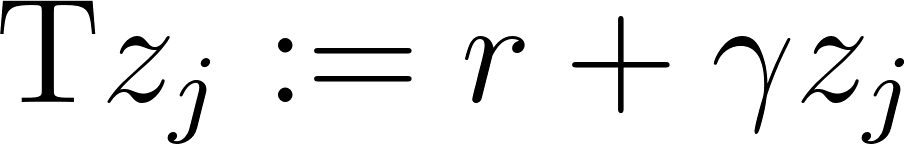
Giá trị phân phối được mô hình hóa bằng một phân phối rời rạc với [](https://www.codecogs.com/eqnedit.php?latex=N%3D51#0), [](https://www.codecogs.com/eqnedit.php?latex=V_%7Bmax%7D%20%3D%2010#0), [](https://www.codecogs.com/eqnedit.php?latex=V_%7Bmin%7D%20%3D%20-10#0) và tập các phần tử hỗ trợ:

[](https://www.codecogs.com/eqnedit.php?latex=z_i%3DV_%7BMIN%7D%2Bi%5CDelta%20z%2C%200%20%5Cle%20i%20%3C%20N#0)

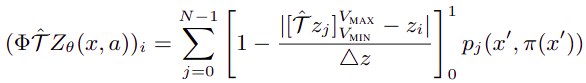
Và

[](https://www.codecogs.com/eqnedit.php?latex=%5CDelta%20z%20%3D%20%5Cfrac%7BV_%7BMAX%7D-V_%7BMIN%7D%7D%7BN-1%7D#0)

Tính cập nhật Bellman cho từng phần tử [](https://www.codecogs.com/eqnedit.php?latex=%20z_j%20#0)

[](https://www.codecogs.com/eqnedit.php?latex=%20%5Cmathrm%7BT%7D%20z_j%20%3A%3D%20r%20%2B%20%5Cgamma%20z_j%20#0)

Tiếp theo là phân phối xác suất của từng phần tử đối với các lân cận của nó. Dẫn đến cập nhật:



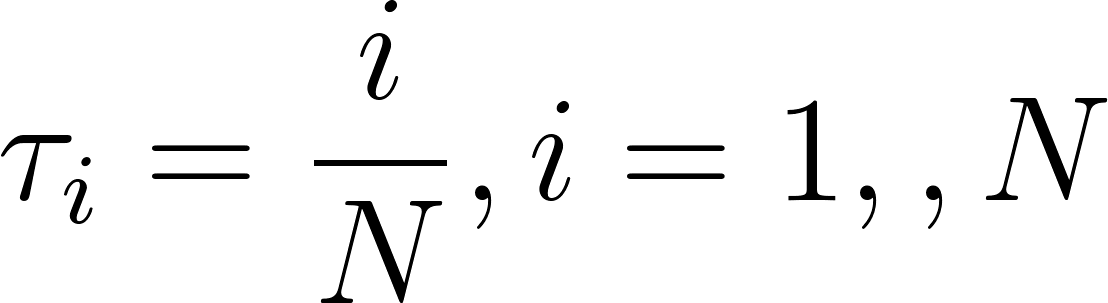
Để giảm thiểu loss, sử dụng số hạng entropy chéo cho phân tán KL:



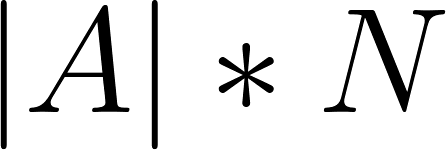
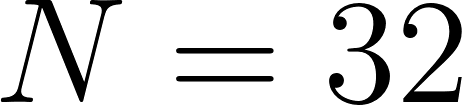
#### 1.4.1.5. QR-DQN (Distributional Reinforcement Learning with Quantile Regression)

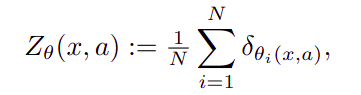
QR-DQN khác với C51 vì nó học trực tiếp từ các giá trị lượng tử. QR-DQN tính toán các lượng tử trả về trên các phân đoạn lượng tử cố định, đồng nhất bằng cách sử dụng hồi quy lượng tử và giảm thiểu tổn thất lượng tử Huber giữa phân phối được cập nhật Bellman và phân phối trả về hiện tại.

So với C51, QR-DQN không bị hạn chế hoặc ràng buộc về giá trị, và do đó cải thiện đáng kể so với C51. Tuy nhiên, cả C51 và QR-DQN đều xấp xỉ hàm phân phối hoặc hàm lượng tử trên các vị trí cố định, cả giá trị hoặc xác suất.

[](https://latex-staging.easygenerator.com/eqneditor/editor.php?latex=%5Ctau_i%20%3D%20%5Cfrac%20%7Bi%7D%7BN%7D%2C%20i%20%3D%201%2C%20%E2%80%A6%20%E2%80%A6%20%E2%80%A6%2C%20N%20#0)

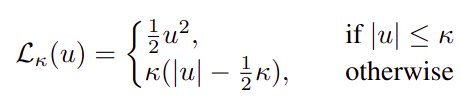
Như đã biết, C51 có 51 giá trị cố định cho phân phối giá trị và học từ xác suất của các giá trị cố định, QR DQN biến đổi các tham số này bằng cách xem xét giá trị xác suất cố định.

Để chuyển từ DQN sang QR-DQN, đầu tiên thay đổi lớp output có độ rộng [](https://www.codecogs.com/eqnedit.php?latex=%7CA%7C*%20N#0), trong đó [](https://www.codecogs.com/eqnedit.php?latex=%20N%3D32#0) là tham số đưa ra số giá trị lượng tử, A là không gian action.



[](https://www.codecogs.com/eqnedit.php?latex=Z_%7B%5Ctheta%7D#0) có thể được xem như là giá trị Q-Value

Thay đổi thứ 2 là thay đổi hàm loss, trong đó có thể là Huber hoặc MSE và thay thế nó bởi hàm Huber loss lượng tử:



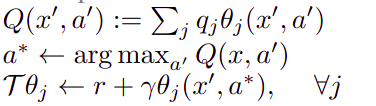
với

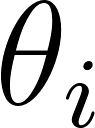


Khi k đặt là 1, u là sai số TD:

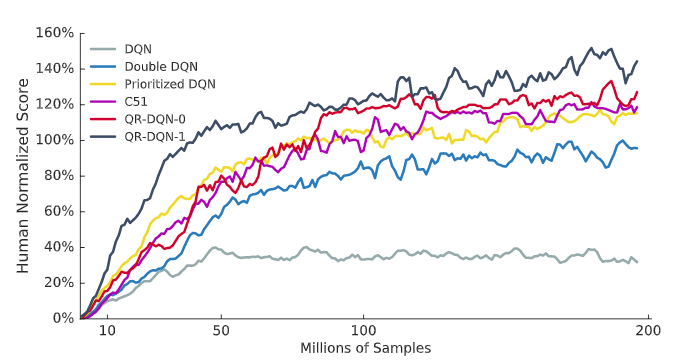


với phân phối Bellman:



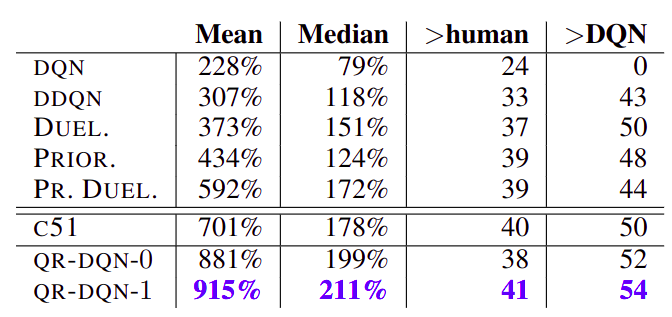
và [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta_i#0) là giá trị ước lượng cho cặp giá trị state-action hiện tại.

Với một số thay đổi nhỏ, QR-DQN không chỉ tốt hơn DQN mà còn hơn 33% điểm trung bình so với C51:



Hình 1.6: Đồ thị biểu diễn các thuật toán QR-DQN

Bảng 1.1: So sánh các thuật toán

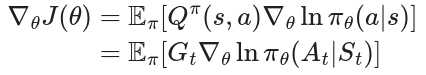


### 1.4.2. Các thuật toán dựa trên policy

#### 1.4.2.1. Reinforce

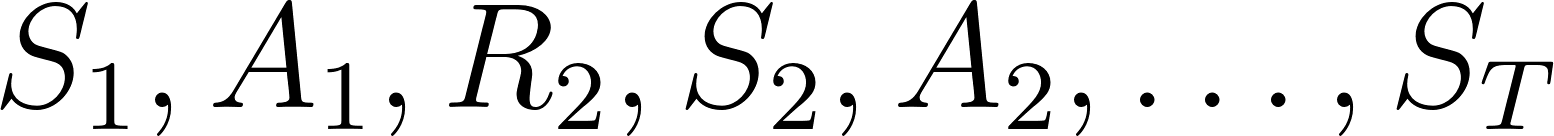
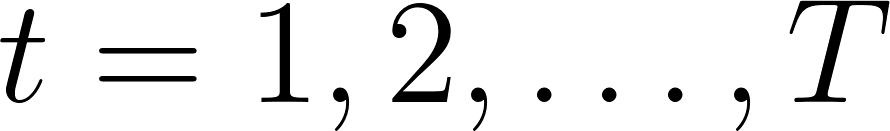
**Reinforce** (Monte-Carlo policy gradient) phụ thuộc vào giá trị ước lược của phương pháp Monte-Carlo sử dụng các mẫu episode để cập nhật tham số policy [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta#0)

REINFORCE cho gradient mẫu bằng với gradient thực tế:



Vì vậy ta có thể xác đinh [](https://www.codecogs.com/eqnedit.php?latex=G_t#0) từ mẫu thực tế và sử dụng để cập nhật policy gradient.

Quá trình diễn ra như sau:

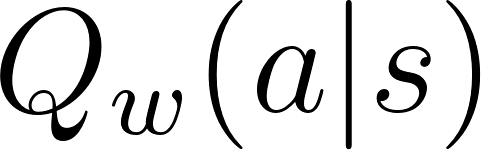
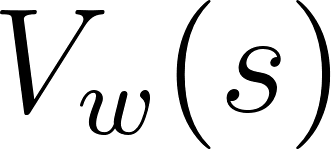
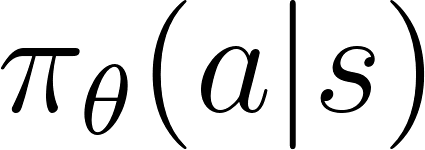
1. Khởi tạo giá trị [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta#0) ngẫu nhiên
2. Sinh quỹ đạo trên policy [](https://www.codecogs.com/eqnedit.php?latex=%5Cpi_%5Ctheta#0): [](https://www.codecogs.com/eqnedit.php?latex=S_1%2C%20A_1%2C%20R_2%2C%20S_2%2C%20A_2%2C%20%5Cldots%2C%20S_T#0)
3. Với mỗi giá trị [](https://www.codecogs.com/eqnedit.php?latex=t%20%3D%201%2C%202%2C%20%5Cldots%2C%20T#0):
   1. Ước lượng giá trị trả về [](https://www.codecogs.com/eqnedit.php?latex=G_t#0)
   2. Cập nhật tham số policy: [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta%20%5Cleftarrow%20%5Ctheta%20%2B%20%5Calpha%20%5Cgamma%5Et%20G_t%20%5Cnabla_%5Ctheta%20ln%20%5Cpi_%5Ctheta%20(A_t%7CS_t)#0)

Một biến thể phổ biến của REINFORCE là trừ đi một giá trị baseline từ [](https://www.codecogs.com/eqnedit.php?latex=G_t#0) để giảm phương sai của ước lượng gradient trong khi vẫn giữ độ chệch.

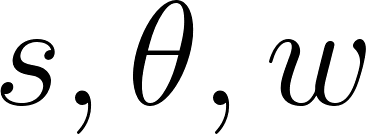
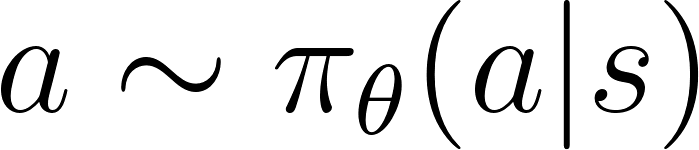
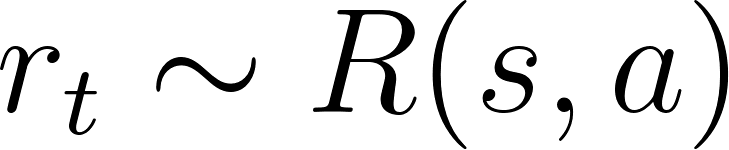
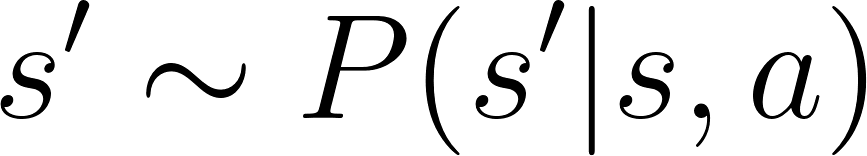
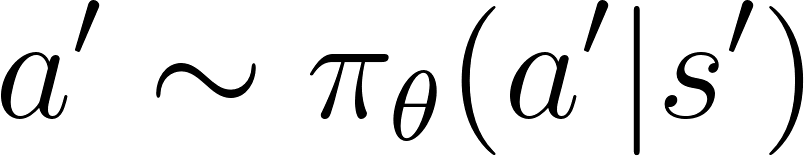
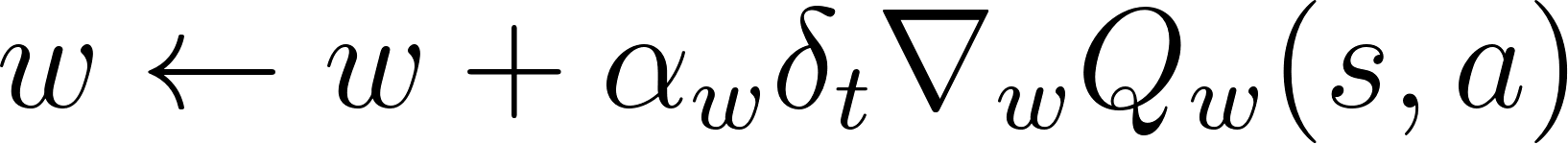
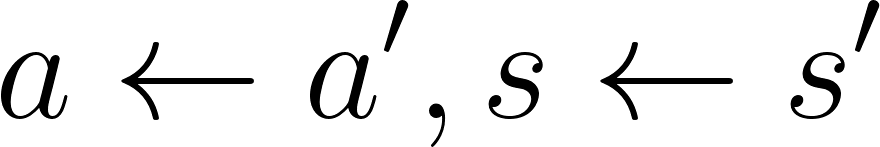
#### 1.4.2.2. Actor-Critic

Hai thành phần chính của policy gradient là mô hình policy và hàm value. Phương pháp Actor-Critic thực hiện học từ hàm value để hỗ trợ policy như giảm phương sai trong phương pháp policy gradient gốc.

**Actor-critic bao gồm 2 mô hình:**

* Critic cập nhật tham số [](https://www.codecogs.com/eqnedit.php?latex=w#0) hàm value và tùy thuộc vào thuật toán, nó có thể là action-value [](https://www.codecogs.com/eqnedit.php?latex=Q_w%20(a%7Cs)#0) hoặc state-value [](https://www.codecogs.com/eqnedit.php?latex=V_w(s)#0)
* Actor cập nhật tham số policy [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta#0) cho [](https://www.codecogs.com/eqnedit.php?latex=%5Cpi_%5Ctheta%20(a%7Cs)#0) theo hướng của Critic

Hoạt động trong một thuật toán action-value actor-critic đơn giản:

1. Khởi tạo [](https://www.codecogs.com/eqnedit.php?latex=s%2C%20%5Ctheta%2C%20w#0) ngẫu nhiên, mẫu [](https://www.codecogs.com/eqnedit.php?latex=a%5Csim%5Cpi_%5Ctheta(a%7Cs)#0)
2. Với mỗi t:
   1. Lấy mẫu phần thưởng [](https://www.codecogs.com/eqnedit.php?latex=r_t%20%5Csim%20R(s%2C%20a)#0) và state tiếp theo [](https://www.codecogs.com/eqnedit.php?latex=s'%5Csim%20P(s'%7Cs%2C%20a)#0)
   2. Lấy mẫu action tiếp theo [](https://www.codecogs.com/eqnedit.php?latex=%20a'%5Csim%5Cpi_%5Ctheta(a'%7Cs')#0)
   3. Cập nhật tham số policy [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta%20%5Cleftarrow%20%5Ctheta%20%2B%20%5Calpha_%5Ctheta%20%20Q_w(s%2C%20a)%20%5Cnabla_%5Ctheta%20ln%20%5Cpi_%5Ctheta(a%7Cs)#0)
   4. Tính sai số TD cho action-value tại t: [](https://www.codecogs.com/eqnedit.php?latex=%5Cdelta_t%20%3D%20r_t%20%2B%20%5Cgamma%20Q_w(s'%2C%20a')%20-%20Q_w(s%2C%20a)#0) và sử dụng nó để cập nhật tham số của hàm action-value: [](https://www.codecogs.com/eqnedit.php?latex=w%20%5Cleftarrow%20w%20%2B%20%5Calpha_w%20%5Cdelta_t%20%5Cnabla_w%20Q_w(s%2C%20a)#0)
   5. Cập nhật [](https://www.codecogs.com/eqnedit.php?latex=a%5Cleftarrow%20a'%2C%20s%5Cleftarrow%20s'#0)

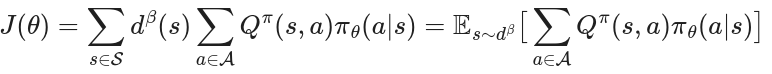
Giá trị học [](https://www.codecogs.com/eqnedit.php?latex=a_%5Ctheta#0) và [](https://www.codecogs.com/eqnedit.php?latex=a_w#0) lần lượt được định nghĩa là tham số policy và tham số hàm value.

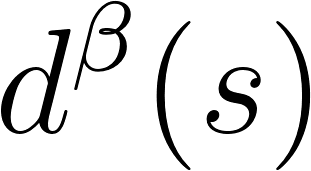
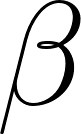
#### 1.4.2.3. Off-Policy Policy Gradient

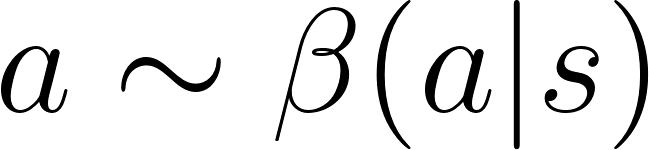
Cả REINFORCE và bản gốc của phương pháp actor-critic đều là on-policy, tức là mẫu huấn luyện được lấy dựa vào policy mục tiêu, rất giống policy ta muốn tối ưu. Phương pháp off policy có một số ưu điểm sau:

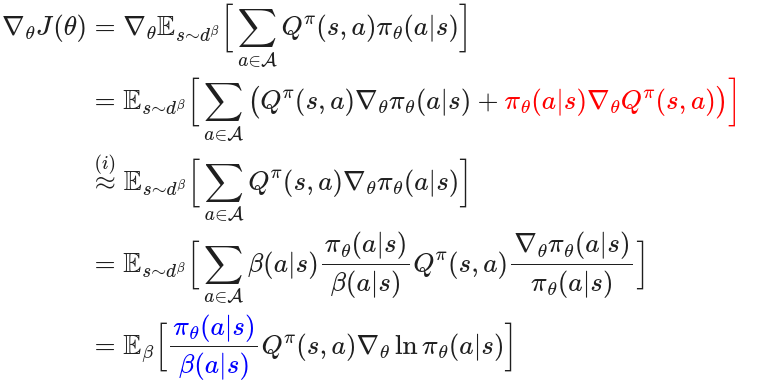
1. Không yêu cầu quỹ đào đầy đủ và có thể sử dụng bất kỳ episode trong quá khứ.
2. Các mẫu thu thập tuân theo policy hành vi khác so với policy đích, mang đến khả năng khai phá tốt hơn.

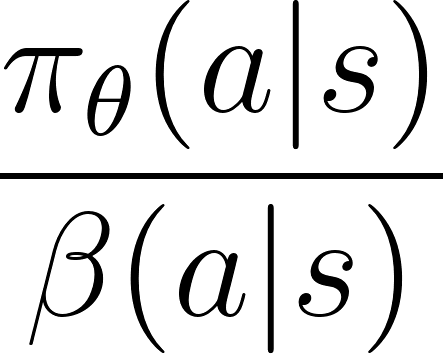
Dưới đây là cách off-policy policy gradient hoạt động. Policy hành vi được sử dụng để thu thập mẫu là một policy đã biết, được ký hiệu là [](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta(a%7Cs)#0). Hàm mục tiêu tổng hợp phần thưởng dựa trên phân phối trạng thái được định nghĩa bởi policy này:



Trong đó [](https://www.codecogs.com/eqnedit.php?latex=d%5E%5Cbeta(s)#0) là phân phối tĩnh của policy hành vi [](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta#0), [](https://www.codecogs.com/eqnedit.php?latex=d%5E%5Cbeta(s)%20%3D%20lim_%7Bt%5Crightarrow%5Cinf%7D%20P(S_t%3Ds%7CS_0%2C%20%5Cbeta)#0) là hàm action-value ước lượng theo policy đích [](https://www.codecogs.com/eqnedit.php?latex=%5Cpi#0)

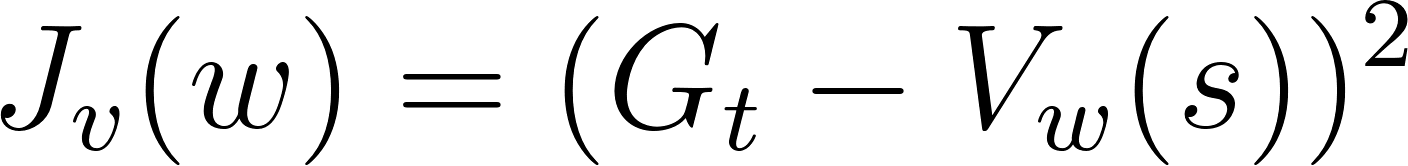
Quá trình huấn luyện được lấy mẫu bởi [](https://www.codecogs.com/eqnedit.php?latex=a%5Csim%5Cbeta(a%7Cs)#0) và có thể viết lại gradient như sau:



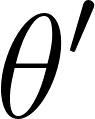
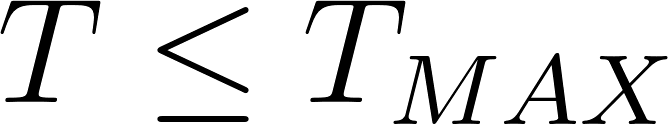
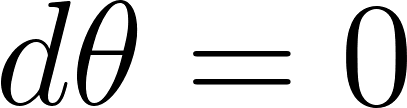
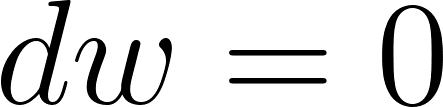
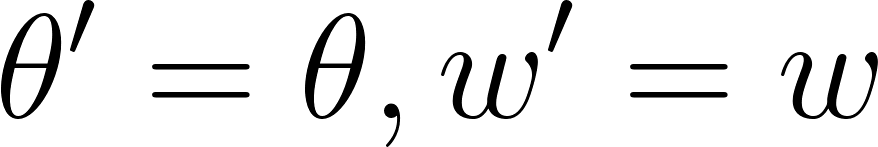
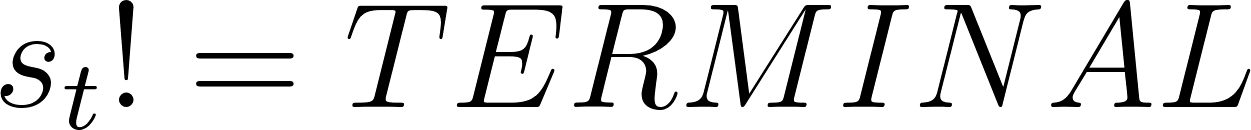
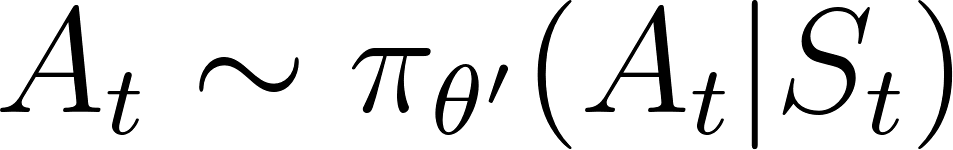
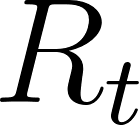
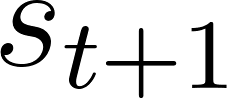
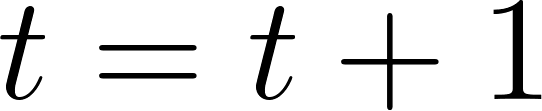
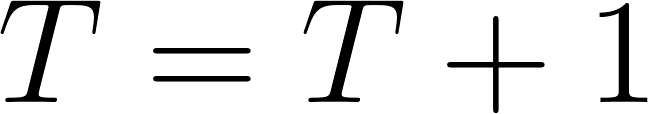
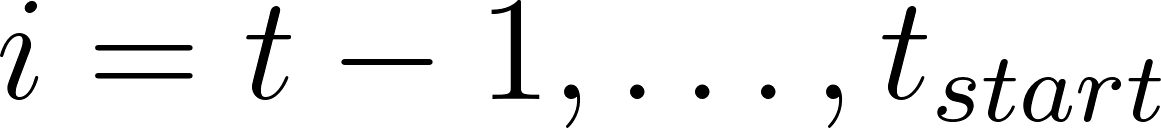
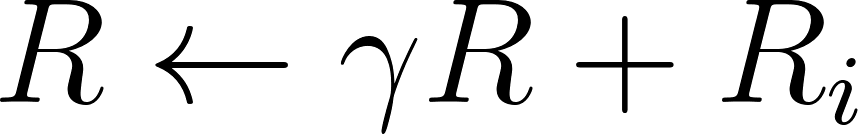
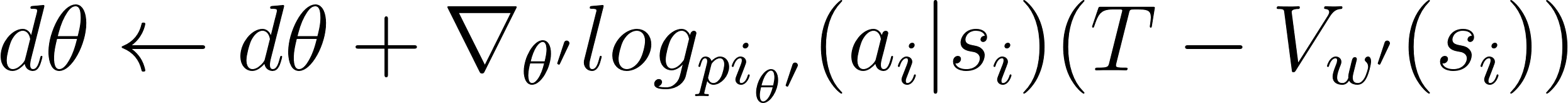
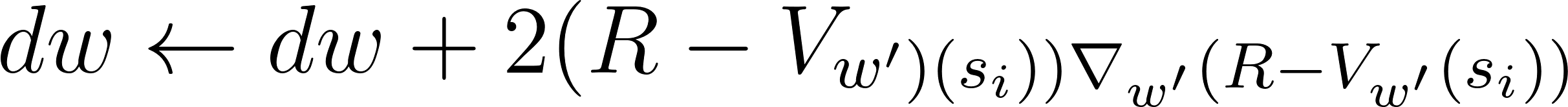
Khi áp dụng policy gradient với phương pháp off-policy, ta có thể dễ dàng thay đổi nó bằng tổng có trọng số và trọng số là tỉ lệ giữa policy đích và policy hành vi: [](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7B%5Cpi_%5Ctheta(a%7Cs)%7D%7B%5Cbeta(a%7Cs)%7D#0)

#### 1.4.2.4. A3C

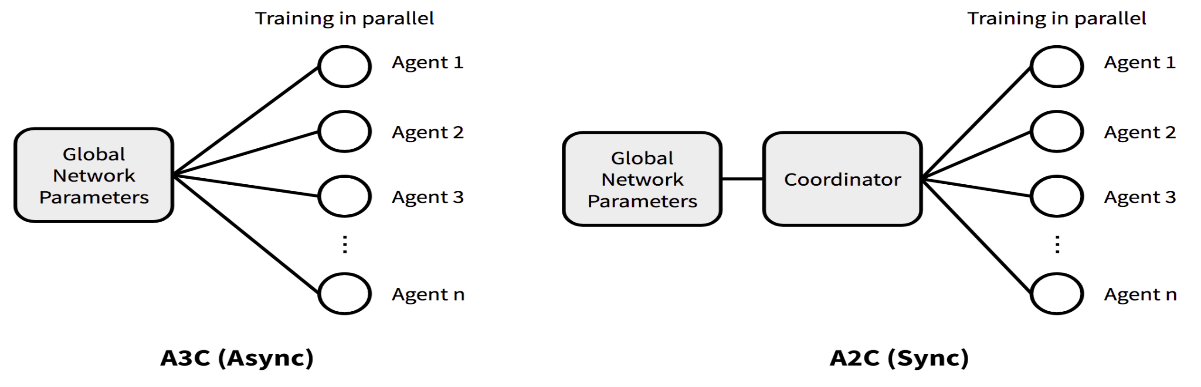
**Asynchronous Advantage Actor-Critic** là một phương pháp policy gradient cổ điển tập trung vào huấn luyện song song.

Critic trong A3C học các hàm value trong khi nhiều actor được huấn luyện song song và được đồng bộ bằng tham số toàn cục. Ví dụ như hàm state-value, hàm loss cho state-value được sử dụng để giảm sai số bình phương [](https://www.codecogs.com/eqnedit.php?latex=J_v(w)%20%3D%20(G_t-V_w(s))%5E2#0) và gradient giảm dần có thể được sử dụng để tìm [](https://www.codecogs.com/eqnedit.php?latex=w#0) tối ưu. Hàm state-value này được sử dụng làm baseline để cập nhật policy gradient.

**Các bước trong thuật toán:**

1. Ta có tham số toán cục [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta#0) và [](https://www.codecogs.com/eqnedit.php?latex=w#0), tương tự là các tham số cục bộ [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta'#0) và [](https://www.codecogs.com/eqnedit.php?latex=w'#0)
2. Khởi tạo thời gian t = 1
3. Với [](https://www.codecogs.com/eqnedit.php?latex=T%5Cle%20T_%7BMAX%7D#0)
   1. [](https://www.codecogs.com/eqnedit.php?latex=d%5Ctheta%20%3D%200#0) và [](https://www.codecogs.com/eqnedit.php?latex=dw%3D0#0)
   2. Đồng bộ các tham số cục bộ với tham số toàn cục: [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta'%20%3D%20%5Ctheta%2C%20w'%3Dw#0)
   3. [](https://www.codecogs.com/eqnedit.php?latex=t_%7Bstart%7D%20%3D%20t#0) và lấy mẫu state bắt đầu [](https://www.codecogs.com/eqnedit.php?latex=s_t#0)
   4. Khi [](https://www.codecogs.com/eqnedit.php?latex=s_t%20!%3D%20TERMINAL#0) và [](https://www.codecogs.com/eqnedit.php?latex=t-t_%7Bstart%7D%20%5Cle%20t_%7Bmax%7D#0)
      1. Chọn action [](https://www.codecogs.com/eqnedit.php?latex=A_t%20%5Csim%20%5Cpi_%7B%5Ctheta'%7D(A_t%7CS_t)#0) và nhận reward [](https://www.codecogs.com/eqnedit.php?latex=R_t#0) và state mới [](https://www.codecogs.com/eqnedit.php?latex=s_%7Bt%2B1%7D#0)
      2. Cập nhật [](https://www.codecogs.com/eqnedit.php?latex=t%3Dt%2B1#0) và [](https://www.codecogs.com/eqnedit.php?latex=T%3DT%2B1#0)
   5. Khởi tạo biến lưu giá trị ước lượng trả về:
   6. Với mỗi giá trị [](https://www.codecogs.com/eqnedit.php?latex=i%20%3D%20t-1%2C%20%5Cldots%2C%20t_%7Bstart%7D#0): [](https://www.codecogs.com/eqnedit.php?latex=R%20%5Cleftarrow%20%5Cgamma%20R%20%2B%20R_i#0), [](https://www.codecogs.com/eqnedit.php?latex=d%5Ctheta%20%5Cleftarrow%20d%5Ctheta%20%2B%20%5Cnabla_%7B%5Ctheta'%7Dlog_%7Bpi_%7B%5Ctheta'%7D%7D(a_i%7Cs_i)(T-V_%7Bw'%7D(s_i))#0) và [](https://www.codecogs.com/eqnedit.php?latex=dw%20%5Cleftarrow%20dw%20%2B%202(R-V_%7Bw')(s_i))%5Cnabla_%7Bw'%7D(R-V_%7Bw'%7D(s_i))#0)
   7. Cập nhật [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta#0) sử dụng [](https://www.codecogs.com/eqnedit.php?latex=d%5Ctheta#0) và [](https://www.codecogs.com/eqnedit.php?latex=w#0) sử dụng [](https://www.codecogs.com/eqnedit.php?latex=dw#0)

#### 1.4.2.5. A2C

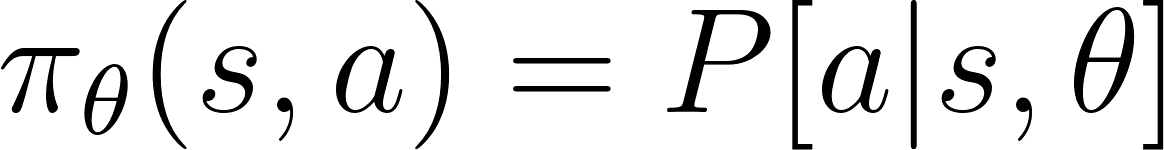
A2C là phiên bản đồng bộ xác định của A3C. Trong A3C, mỗi agent đồng bộ với tham số toàn cục độc lập với nhau, có khả năng dẫn đến các agent thực hiện phiên bản policy khác dẫn đến cập nhật không được tối ưu. Để giải quyết sự không nhất quán này, A2C chờ tất cả actor song song hoàn thành công việc trước khi cập nhật tham số toàn cục, sau đó vòng lặp tiếp theo các actor bắt đầu từ một policy. A2C cho thấy việc sử dụng GPU một cách hiệu quả hơn và hoạt động tốt hơn với việc xử lý theo lô lớn trong khi mang lại hiệu năng giống hoặc tốt hơn A3

Hình 1.7: So sánh A3C và A2C

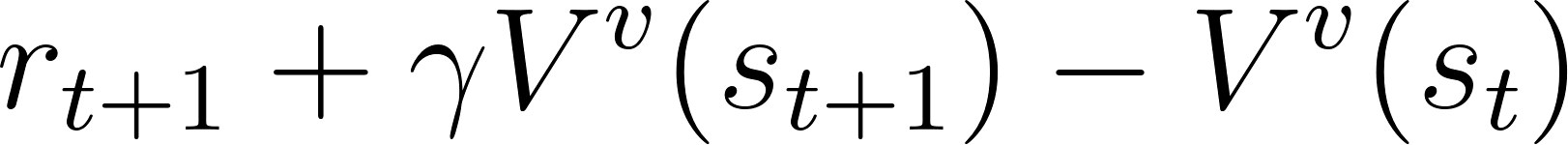
#### 1.4.2.6. DDPG (Deep Deterministic Policy Gradient)

Mặc dù DQN thành công ở nhiều vấn đề, không gian action vẫn bị rời rạc. Trong một số tác vụ như điều khiển vật lý, không gian action là liên tục. Nếu rời rạc hóa không gian action quá tốt sẽ dẫn đến không gian này trở nên quá lớn.

DDPG dựa trên kiến trúc actor-critic với 2 thành phần Actor và Critic. Actor được sử dụng để hiệu chỉnh [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta#0) cho hàm policy (quyết định action tốt nhất cho state).

[](https://www.codecogs.com/eqnedit.php?latex=%5Cpi_%7B%5Ctheta%7D(s%2C%20a)%20%3D%20P%5Ba%7Cs%2C%20%5Ctheta%5D#0)

Critic được sử dụng để đánh giá policy function của actor dựa trên lỗi Temporal Difference:

[](https://www.codecogs.com/eqnedit.php?latex=r_%7Bt%2B1%7D%20%2B%20%5Cgamma%20V%5Ev(s_%7Bt%2B1%7D)-V%5Ev(s_t)#0)

trong đó [](https://www.codecogs.com/eqnedit.php?latex=v#0) là policy được chọn bởi actor.

## 1.5. Các thư viện

### 1.5.1. OpenAI Baselines

Thư viện Baselines hỗ trợ các thuật toán RL như A2C, ACER, ACKTR, DDPG, DQN, GAIL, HER, PPO, TRPO. Cho phép người dùng huấn luyện model và hỗ trợ logger mô phỏng quá trình huấn luyện. Sau huấn luyện, model có thể được lưu để sử dụng lại. Nhược điểm là thư viện AI Baseline không có đầy đủ tài liệu để theo dõi.

### 1.5.2. Stable Baselines

Stable Baselines là một bản fork của thư viện OpenAI Baselines với nhiều cải tiến. Stable Baseline tái cấu trúc lại OpenAI Baselines cho một cấu trúc và giao diện quen thuộc cho các thuật toán.

Ngoài tất cả thuật toán của OpenAI, Stable Baselines cung cấp 2 thuật toán nữa là Soft Actor-Critic(SAC) và Twin Delayed DDPG(TD3) và hỗ trợ Tensorboard.

### 1.5.3. Keras-RL

Keras-RL là một thư viện học sâu hỗ trợ các thuật toán RL hiện đại. Nó hỗ trợ các thuật toán như DQN, DDPG, NAF, CEM, SARSA và có tài liệu giải thích rõ ràng.

### 1.5.4. TF Agents

TF Agents là một thư viện tensorflow cho Reinforcement Learning cung cấp nhiều thành phần RL dễ sử dụng cũng như dễ thay đổi tùy theo nhu cầu sử dụng. Nó cung cấp các thuật toán RL sau: DQN, DDQN, DDPG, TD3, REINFORCE, PPO, SAC. TF Agents có thể tích hợp cho các môi trường RL như OpenAI Gym, DeepMind-control và Atari.

### 1.5.5. PyQlearning

PyQlearning là một thư viện RL tập trung vào Q-learning. Nó hỗ trợ cả Deep Q-learning và multi-agent Deep Q-learning.

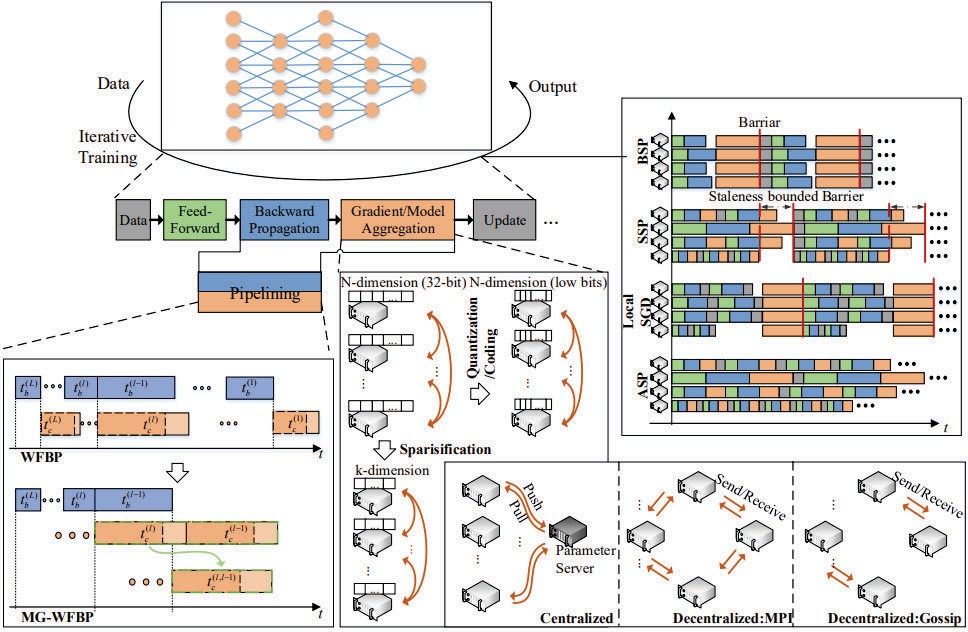
### 1.5.6. Tensorforce

Tensorforce là thư viện học sâu tăng cường mã nguồn mở được xây dựng trên nền Tensorflow. Thư viện này được thiết kế dạng modular cho khả năng dễ sử dụng. Nó cung cấp một số thuật toán RL như DQN, Policy Gradient, Actor-Critic, … thư viện có hỗ trợ Tensorboard cho việc mô phỏng.

# 2. HỌC PHÂN TÁN

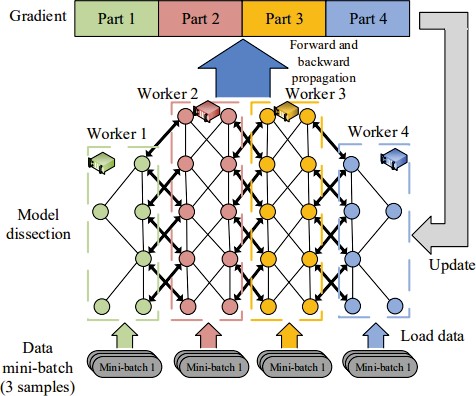
Học sâu đem lại hiệu quả vượt trội cho ngành trí tuệ nhân tạo những năm trở lại đây. Phương pháp huấn luyện mạng học sâu được các nhà nghiên cứu và kĩ sư áp dụng vào giải quyết nhiều bài toán thực tế, ở nhiều lĩnh vực như thị giác máy tính, xử lý ngôn ngữ tự nhiên, nhận diện giọng nói và nhiều hơn thế. Tuy nhiên, để đạt được độ chính xác cao, cũng như hiệu quả vượt trội, học sâu cần tăng về cả số lượng và chất lượng của dữ liệu. Điều này dẫn tới bài toán về xử lý độ phức tạp của kích thước mô hình và xử lý dữ liệu lớn trong quá tình huấn luyện mạng, bên cạnh đó cần đảm bảo năng lực tính toán đơn lẻ và giảm thời gian tiêu tốn. Ví dụ như huấn luyện mô hình ResNet-50 (trong 90 epochs) trên tập dữ liệu ImageNet cùng GPU mới nhất của Nvidia là Tesla V100 đã phải mất tới 2 ngày.

Nhằm giảm thiểu thời gian huấn luyện mạng, tận dụng nguồn dữ liệu từ nhiều bên tham gia, các phương pháp học sâu phân tán được đưa ra nhằm tận dụng tối đa cơ sở hạ tầng phần cứng và phần mềm. Các phương pháp huấn luyện mô hình phân tán được đưa ra, và càng trở nên phổ biến hơn trong những năm gần đây. Phương pháp huấn luyện trên môi trường phân tán không những tận dụng được toàn bộ cơ sở vật lý tính toán và giảm thiểu thời gian huấn luyện đáng kể. Ngoài ra phương pháp huấn luyện phân tán cũng làm cho dữ liệu cục bộ không cần phải chia sẻ đến mảy chủ tập trung do đó giúp loại bỏ các nguy cơ về rò rỉ dữ liệu trực tiếp. Về cơ bản, việc huấn luyện mô hình phân tán có thể nhóm thành 2 phương pháp tính toán song song chính: song song mô hình (model parallelism) và song song dữ liệu (data parallelism).



Hình 2.1: Tổng quan về huấn luyện phân tán mạng học sâu

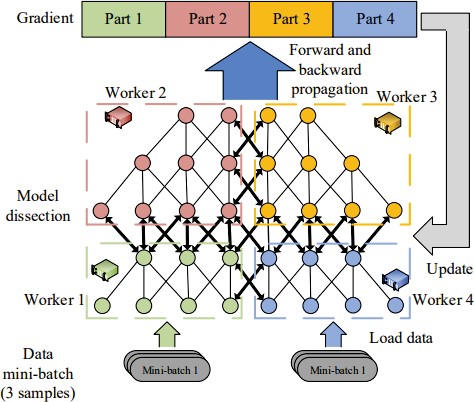
Song song hóa mô hình (Model Parallelism) là phương pháp chia các tham số mô hình cho các thành viên tham gia. Các thành viên này giữ các tham số khác nhau hoặc các lớp khác nhau của mô hình. Chúng ta có thể thấy cách chia này trong Hình 2.2 và Hình 2.3.



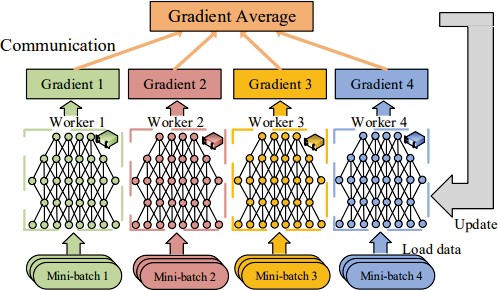
Hình 2.2: Mô hình huấn luyện phân tán chia theo lớp

Trong Hình 2.2. ta có thể thấy rằng các tham số trong mạng nơ-ron được phân tán cho các thành viên dưới dạng chiều dọc, mỗi bên sẽ đóng góp tài nguyên để huấn luyện một phần của mạng lớn từ lớp đầu cho đến lớp kết thúc. Ngược lại, trong mô hình ở Hình 2.3, các lớp cũng được chia cho các thành viên tham gia. Mỗi thành viên đóng vai trò huấn luyện một số lớp thuộc một phần nhất định của mô hình tổng quát.

Dựa vào sự tương quan phụ thuộc lẫn nhau của các nơ ron trọng mô hình học sâu, các đơn vị thành viên trao đổi kết quả đầu ra với nhau trước khi thực hiện tiếp tục quá trình tính toán ở lớp tiếp theo. Một trong những ưu điểm của việc song song hóa tính toán, huấn luyện mô hình chính là việc huấn luyện mô hình có kích thước lớn trở nên khả thi cho mỗi đơn vị huấn luyện thành viên tham gia, và mỗi đơn vị thành viên này nắm giữ một tập con của mô hình và yêu cầu bộ nhớ tính toán từ rất lớn giờ đây đã được chia nhỏ cho các đơn vị thành viên. Tuy nhiên, phương pháp này còn tồn đọng một số vấn đề, có thể kể đến như là mất cân bằng khối lượng tham số và việc chênh lệch tính toán giữa các lớp trong mô hình học sâu. Hiện nay, việc thiết kế thuật toán cài đặt song song hóa mô hình vẫn còn là một thách thức rất lớn.



Hình 2.3: Mô hình huấn luyện phân tán chia theo node

Ngoài ra, khi song song hóa mô hình, ta có thể thấy rằng, các dữ liệu đầu vào là tập trung và giống nhau giữa các node tham gia. Vì vậy, mô hình này không đóng nhiều vai trò trong việc xây dựng và huấn luyện các mô hình học sâu phân tán có đảm bảo tính riêng tư. Thông thường, các mô hình đảm bảo tính riêng tư được thiết kế theo mô hình song song hóa dữ liệu. Tức là mỗi bên tham gia có một bộ dữ liệu riêng và việc huấn luyện cục bộ xảy ra trên bộ dữ liệu này. Chúng ta có thể tham khảo Hình 2.4, trong đó tham số mô hình được thực hiện huấn luyện bởi tất cả các đơn vị tính toán thành viên. Trong một lượt tính toán, mỗi tiến trình tính toán bởi mỗi đơn vị thành viên sẽ thực hiện việc tính toán và xử lý một phần nhỏ dữ liệu do thành viên đó sở hữu nhằm cập nhật giá trị gradient địa phương. Sau đó, các gradient hoặc tham số địa phương cục bộ này được trao đổi với các đơn vị thành viên khác trước khi cập nhật để tạo ra tham số của mô hình chính.

Hình 2.4: Phương pháp song song hóa dữ liệu

# TÀI LIỆU THAM KHẢO

1. Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A briefreview. Computational intelligence and neuroscience, 2018.
2. Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. Ieee Computational intelligenCe magazine, 13(3):55–75, 2018.
3. Andrew Boles and Paul Rad. Voice biometrics: Deep learning-based voiceprint authentication system. In 2017 12th System of Systems Engineering Conference (SoSE), pages 1–6. IEEE, 2017 Valentina Emilia Balas, Sanjiban Sekhar Roy, Dharmendra Sharma, and Pijush Samui. Handbook of deep learning applications, volume 136. Springer, 2019.
4. Honglak Lee, Roger B. Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009, volume 382 of ACM International Conference Proceeding Series, pages 609–616. ACM, 2009.
5. YuxinWang, QiangWang, Shaohuai Shi, Xin He, Zhenheng Tang, Kaiyong Zhao, and Xiaowen Chu. Benchmarking the performance and energy efficiency of ai accelerators for ai training. In 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), pages 744–751. IEEE, 2020.
6. Zhenheng Tang, Shaohuai Shi, Xiaowen Chu, Wei Wang, and Bo Li. Communication-efficient distributed deep learning: A comprehensive survey. arXiv preprint arXiv:2003.06307, 2020.