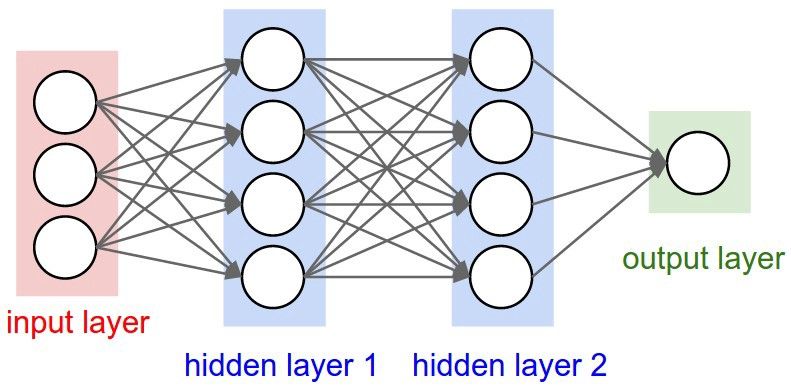
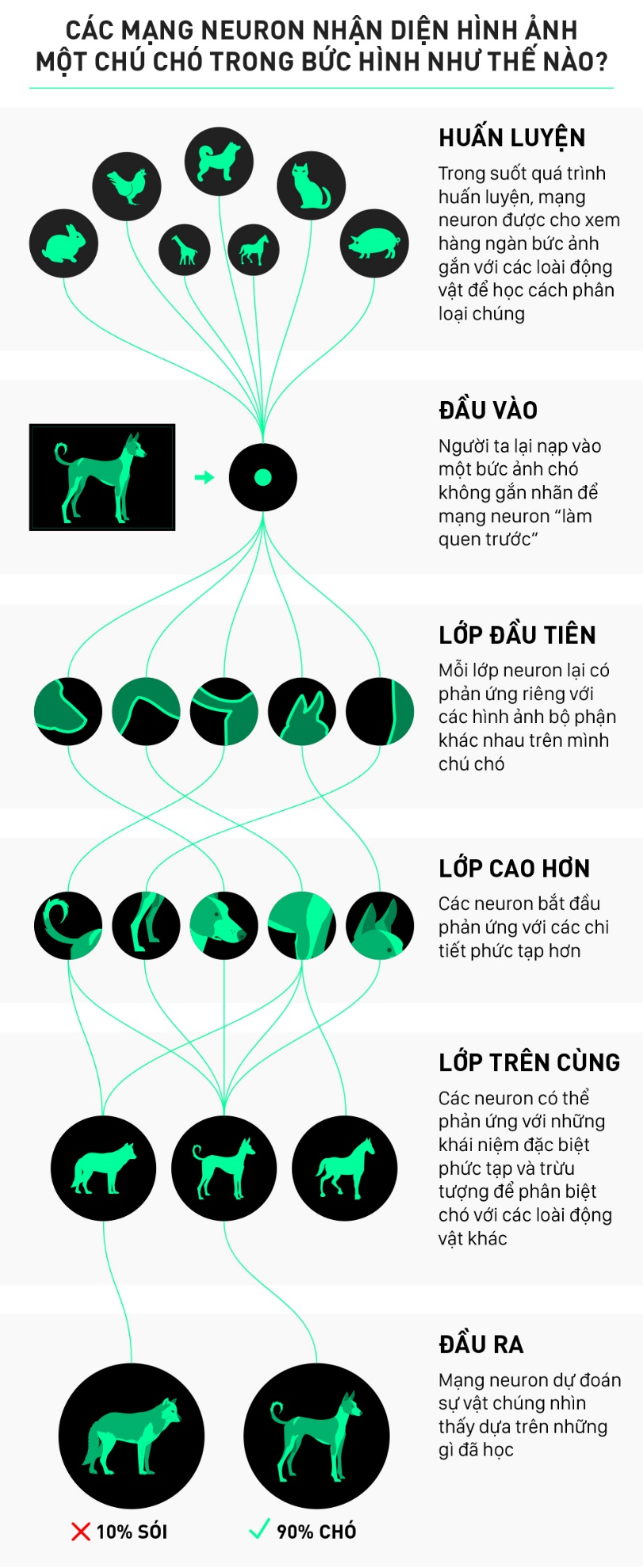
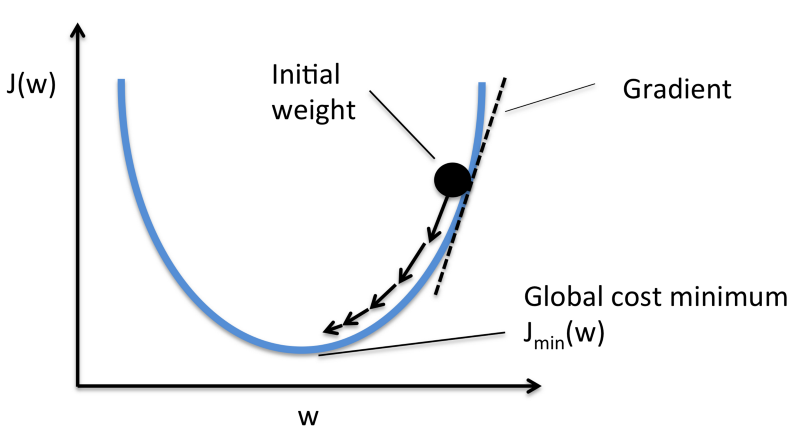
**Exploration with Exemplar Models for Deep Reinforcement Learning – Phương pháp khám phá cùng các mô hình mẫu trong Deep Reinforcement Learning**

Introduction  
1) Khái niệm Deep Learning:  
Deep Learning là một phương pháp của Học máy – Machine Learning. Nó cho phép chúng ta huấn luyện một AI có thể dự đoán được các đầu ra dựa vào một tập các đầu vào. Cả hai phương pháp có giám sát (supervised Learning) và không giám sát (unsupervised learning) đều có thể sử dụng để huấn luyện.  
Deep learning giúp máy tự huấn luyện chính nó cách thực hiện các tác vụ như: nhận diện giọng nói bằng cách cho các lớp thần kinh nhân tạo trong mạng neuron sâu tiếp xúc với một lượng dữ liệu khổng lồ.  
2) Mạng nơ ron:  
Deep Learning sử dụng một mạng nơ ron để bắt chước trí thông minh của động vật ( giống hành động của vật).  
Cũng giống động vật, bộ não của AI cũng có các nơ ron. Chúng được biểu diễn bằng các vòng tròn. Các nơ ron này đều đã được liên kết.  
  
Các nơ ron được chia làm 3 loại: Input layer, Hidden layer, và output layer  
a) Input Layer: nhận các dữ liệu đầu vào, Ví dụ đưa vào: độ ẩm, sức gió, nhiệt độ, ..  
b) Hidden layer:  thực hiện các phép tính toán cho các đầu vào. Thử thách lớn nhất trong việc tạo mạng nơ ron là quyết định số lượng các hidden layer này, cũng như số các nơ ron cho mỗi layer – “ Từ deep trong deep learning chỉ đến việc có nhiều hơn một hidden layer”.  
b) Output layer: Giá trị đầu ra : Trời có mưa hay không,…  
3) Cách máy móc tính toán :  
Mỗi một kết nối giữa nơ ron được liên kết với một trọng số (weight). Trọng số này chỉ rõ ra tầm quan trọng của giá trị đầu vào. Trọng số khởi tạo được chọn ngẫu nhiên.  
Ví dụ: Khi dự đoán thời tiết, thì độ ẩm là quan trọng nhất -> mạng nơ ron liên kết của độ ẩm có trọng số lớn  
Mỗi một nơ ron sẽ có một Hàm kích hoạt ([Activation Function](https://en.wikipedia.org/wiki/Activation_function)). Các hàm này sẽ là các thuật toán như: softmax, gaussian.... Một trong những mục đích của nó là để chuẩn hóa output từ nơ ron. Khi một tập dữ liệu input được truyền qua tất cả các layer của mạng nơ ron, nó sẽ trả về dữ liệu đầu ra thông qua output layer.  
4) Huyến luyện nơ ron – Huyến luyện Neural Network  
Để huấn luyện AI, chúng ta cần đưa cấc đầu vào từ tập dữ liệu, sau đó so sánh với đầu ra của nó với đầu ra của tập dữ liệu mẫu. Khi AI chưa được huấn luyện, đầu ra của nó sẽ có thể bị sai.  
Khi ta đã hoàn thành với tập dữ liệu, ta có thể tạo một hàm hiển thị độ sai của đầu ra của AI so với đầu ra thực tế. Hàm này được gọi là **Cost Function** . Một cách hiểu đơn giản rằng, ta muốn Cost Function của ta sẽ trả về 0, khi đó đầu ra của AI cũng sẽ giống với đầu ra thực tế từ tập dữ liệu.  
  
5) Cách giảm Cost Function:  
Để giảm giá trị Cost Function ta phải thay đổi trọng số giữa các nơ ron -> bằng cách sử dụng công nghệ gọi là Gradient Descent  
6) Gradient Descent  
Gradient Descent là công nghệ cho phép ta tìm giá trị nhỏ nhất của một hàm. Trong trường hợp của ta, chúng ta đang tìm giá trị nhỏ nhất cho cost function. Nó hoạt động bằng cách thay đổi một giá trị rất nhỏ cho trọng số sau mỗi lần lặp trong tập dữ liệu. Bằng cách tính toán đạo hàm của cost function ở một tập trọng số, ta có thể tìm được hướng của cực tiểu.  
  
Để giảm tối đa cost function, bạn phải lặp rất nhiều lần trong tập dữ liệu. Đây là lý do tại sao phải cần một khả năng tính toán rất lớn. Cập nhật trọng số sử dụng gradient descent được hoàn thành một cách tự động.  
7) Một số ví dụ về Deep Learning hiện nay  
Những trợ lý ảo giọng nói như Alexa của Amazon, Siri của Apple, Cortana của Microsoft cùng các hệ thống nhận diện giọng nói có mặt trên hầu như mọi sản phẩm của Google đang đồng loạt nở rộ trên nhiều nền tảng, giúp người dùng thực hiện nhiều tác vụ khác nhau. Ở phía bên kia địa cầu, gã khổng lồ tìm kiếm Baidu cũng tỏ ra không hề kém cạnh với thống kê cho thấy người dùng các sản phẩm của công ty đã sử dụng nhận diện giọng nói nhiều gấp ba lần chỉ trong vòng 18 tháng qua.  
8) Những điểm còn hạn chế.  
Hiện nay hầu hết các phương pháp ước tính mới được đưa ra dựa vào việc xây dựng các mô hình mang tính tiên đoán hoặc dự đoán đều rõ ràng sự phân bố trên quan sát hiện tại hoặc tiếp theo. Còn khi quan sác có tính phứt tạp, như hình ảnh thô, thì các mô hình khó đạo tạo.  
9) Bài toán ví dụ:  
Giả sử chúng ta cần xây dựng AI Bot để chơi game Flappy bird.  
Chúng ta cần xây dựng mạng nơron với input là hình ảnh game, output là Tap hoặc không Tap. Mục tiêu là giúp con chim đi được càng xa càng tốt.  
Rõ ràng với mỗi trường hợp khác nhau, AI Bot phải quyết định nên Tap hay không? Nó có vẻ phụ thuộc nhiều vào trực giác, cứ mỗi khi con chim rớt là phải Tap để nó bay lên https://www.facebook.com/images/emoji.php/v9/fa5/1.5/16/1f642.png=) Làm sao AI có thể nhận định được là tình huống này nên Tap hay không.  
Học tăng cường sẽ tập trung giải quyết vấn đề này.

Hầu hết các vấn đề của học tăng cường đều xoay quanh các khái niệm Trạng thái (State), hành động (action) và Phần thưởng (Reward).   
Trạng thái (State) thể hiện tình trạng của game tại thời điểm nào đó. Thường thì các game có tập trạng thái rất rất lớn.  
Hành động (Action) là tập các hành động cho phép tương tác vào Game.  
Phần thưởng (Reward) thường gắn với Score của game, là mục tiêu của người chơi khi chơi game.  
Mô hình quá trình chơi game căn bản như sau:  
Game có trạng thái ban đầu ---> tương tác Action ---> Game chuyển trạng thái mới và trả ra phần thưởng tương ứng (Reward).

Quay lại game Flappy bird, ta thấy trạng thái là vị trí hiện thời của con chim và vị trí của các cột. Hành động là Tap hoặc không Tap. Phần thưởng là số lượng cột mà con chim có thể vượt qua.  
Khi bạn nhận Phần thưởng càng cao nghĩa là bạn đang làm càng tốt (chim bay càng xa, vượt càng nhiều cột). Mặc dù, đôi khi Phần thưởng không được thể hiện ngay tức thì, đôi lúc Phần thưởng sẽ trễ vài ms, vài giây thậm chí vài giờ sau khi bạn action. Điều này nảy sinh thách thức đầu tiên của Học tăng cường gọi là   
"credit assignment problem" (không biết dịch sang tiếng Việt ntn). Làm sao chúng ta biết được chính xác Hành động nào dẫn đến Phần thưởng đó? Một trong những phương pháp thường dùng là "discounted future rewards". Ý tưởng chính của phương pháp này là sử dụng một toán tử giảm γ (gamma):  
Rt=rt+γrt+1+γ2rt+2+...+γn−1rn  
viết một cách đệ quy thì như thế này:  
Rt=rt+γRt+1  
Trong đó γ thường được lấy là 0.9 hoặc 0.99, nghĩa là Phần thưởng cho các Hành động ở tương lai càng xa thì càng bị giảm.  
Hai câu hỏi có thể được đặt ra là:  
1. Tại sao Phần thưởng trong tương lai lại quan trọng? Nếu ta chỉ quan tâm tới Phần thưởng tại hiện tại (γ=0) thì sao?  
Trả lời: Đầu tiên, Phần thưởng tương lai được dự đoán sẽ cho AI khả năng nghĩ "trước 3 bước", trong game ai tính được trước nước đi đã giành phần thắng cao hơn rồi.  
Thứ hai, trong hầu hết các game, nước đi đầu tiên thường quan trọng hơn những nước tiếp theo, đi sai một phát là đi bụi liền, các nước đi có liên quan với nhau. Vì thế cho nên Phần thưởng cho các nước đi đầu sẽ bao gồm Phần thưởng cho các nước đi kế tiếp.  
Cuối cùng, Phần thưởng trong game thường đến End game mới biết vì thế nếu không có γ thì toàn bộ nước đi trước nước đi cuối cùng đều không có ý nghĩa, vì chỉ đi nước cuối mới biết Thắng-Thua.  
2. Vậy nếu ta chỉ quan tâm tới Tổng cộng Phần thưởng (tức là γ = 1)?  
Càng đi nhiều nước đi trong game, ta càng nhận được những trạng thái không chắc chắn, nghĩa là Trạng thái của game là ngẫu nhiên, càng đi xa ta càng không thể chắc chắn về Phần thưởng sẽ nhận được do đó nên giảm bớt sự ảnh hưởng của Phần thưởng không chắc chắn.  
Ví dụ khi chơi game Tiến lên, chẳng có gì đảm bảo rằng ta ra 1 quân bài tại thời điểm hiện tại sẽ giúp ta có lợi thế giống như lợi thế khi ta ra quân bài đó trong quá khứ. Càng chơi lâu thì tỷ lệ giống quá khứ càng giảm.

Còn vấn đề khác nữa là, giả sử mình có chiến lược giúp AI nhận định được tình huống và luôn nhận được phần thưởng tốt tại mỗi trạng thái game. Câu hỏi là liệu ta có nên cứ tiếp tục áp dụng chiến lược đó vì nó vẫn đang mang lại kết quả tốt, hay ta nên thử các chiến lược mới biết đâu có được chiến lược tốt hơn nữa thì sao? Điều này nảy sinh ra vấn đề "Khám phá và khai thác" (Exploration and Exploitation)

Deep Q Network (DQN) là một trong những giải thuật Học tăng cường nổi tiếng gần đây, ý tưởng cốt lõi của giải thuật này là Q-learning (Watkins and Dayan, 1992). Giải thuật Q-Learning kinh điển dựa trên Hàm xấp xỉ Q(st,at)=𝔼[Rt‖st,at]Q(st,at)=E[Rt‖st,at], Hàm này sẽ dự đoán Phần thưởng tương lai cao nhất khi ta thực hiện một hành động a tại trạng thái s. Trong giải thuật gốc, Hàm xấp xỉ Q này được thiết kế là một Bảng gọi là Q-Table, trong đó Hàng là các trạng thái có thể có, cột là các Hành động trong game. Trong quá trình học, bảng này sẽ lưu lại các giá trị Phần thưởng tương lai cao nhất cho mỗi hành động tại 1 trạng thái nhất định.  
Tuy nhiên với thiết kế này khi áp dụng cho game gặp phải vấn đề về lưu trữ Trạng thái trong Q-Table. Tập trạng thái quá lớn để lưu trữ hết trên RAM. Do đó Deep Q-Network ra đời, DQN sẽ thay thế Q-table bằng mạng Nơ ron đủ tốt để có thể dự đoán Phần thưởng cho mỗi hành động tại một trạng thái nhất định. Khi mạng Nơ ron này đủ tốt nó cho phép ta tính toán được Phần thưởng tốt nhất khi thực hiện một hành động tại một Trạng thái bất kỳ. Hàm xấp xỉ lúc này được gọi là Phương trình Bellman như sau:  
Q(st,at)=rt+γmaxat+1Q(st+1,at+1)  
Trong đó, S: Trạng thái  
a: Hành động thực hiện  
r: Phần thưởng hiện tại  
gamma: Toán tử giảm  
Để mạng Nơ ron hội tụ được ta cần hàm lỗi (Loss) để điều chỉnh trọng số mạng:  
L=(r+γmaxat+1Q(st+1,at+1)−Q(st,at))2

10) Phương pháp khám phá - Exprotaion  
Deep reinforcement learning algorithms have been shown to learn complex tasks using highly general policy classes. However, sparse reward problems remain a significant challenge. Exploration methods based on novelty detection have been particularly successful in such settings but typically require generative or predictive models of the observations, which can be difficult to train when the observations are very high-dimensional and complex, as in the case of raw images. We propose a novelty detection algorithm for exploration that is based entirely on discriminatively trained exemplar models, where classifiers are trained to discriminate each visited state against all others. Intuitively, novel states are easier to distinguish against other states seen during training. We show that this kind of discriminative modeling corresponds to implicit density estimation, and that it can be combined with countbased exploration to produce competitive results on a range of popular benchmark tasks, including state-of-the-art results on challenging egocentric observations in the vizDoom benchmark.