TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**NHẬP MÔN HỌC MÁY**

Người hướng dẫn : **Thầy Lê Anh Cường**

Người thực hiện : **Định Thị Ngọc Phượng – 52100923**

**Đinh Phú Quốc – 52100927**

**Cao Đăng Tình – 52100936**

**Lớp : 21050301**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

**LỜI CẢM ƠN**

Đầu tiên, nhóm chúng em xin phép gửi lời các thầy cô khoa Công Nghệ Thông Tin của trường Đại học Tôn Đức Thắng nói chung và đặc biệt là thầy Lê Anh Cường nói riêng. Nhóm chúng em xin cảm ơn thầy vì trong suốt thời gian học tập vừa qua thầy đã truyền tải cho chúng em nhưng kiến thức về môn học Nhập Môn Học Máy cũng như là các kiến thức để giúp chúng em hoàn thành được bài báo cáo này. Có thể trong quá trình làm bài cáo này sẽ phát sinh ra những sự cố không mong muốn, mong thầy có thể góp ý để chúng em có thể hoàn thành bài báo cáo một cách tốt nhất và tránh mắc phải những sự cố tương tự ở các môn học tiếp theo.

***Chúng em cảm ơn thầy rất nhiều ạ!***

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của Giảng viên Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

**Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 10  tháng 10  năm  2023*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

***Đinh Thị Ngọc Phượng***

***Đinh Phú Quốc***

***Cao Đăng Tình***

**PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày     tháng   năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày     tháng   năm

(kí và ghi họ tên)

**MỤC LỤC**

[1 Trình bày nghiên cứu và đánh giá về các vấn đề : 7](#_Toc154054285)

[1.1 Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình : 7](#_Toc154054286)

[1.1.1 Gradient Descent (GD): 7](#_Toc154054287)

[1.1.2 Stochastic Gradient Descent (SGD): 8](#_Toc154054288)

[1.1.3 Mini-batch Gradient Descent: 10](#_Toc154054289)

[1.1.4 Momentum: 12](#_Toc154054290)

[1.1.5 Adagrad: 14](#_Toc154054291)

[1.1.6 RMSprop: 16](#_Toc154054292)

[1.1.7 Adam: 18](#_Toc154054293)

[2 Tìm hiểu về Continual Learning và Test Production: 21](#_Toc154054294)

[2.1 Continual Learning: 21](#_Toc154054295)

[2.1.1 Forgetting và Catastrophic Interference: 22](#_Toc154054296)

[2.1.2 Plasticity và Stability: 24](#_Toc154054297)

[2.1.3 Knowledge Consolidation: 27](#_Toc154054298)

[2.1.4 Transfer Learning và Fine-tuning: 29](#_Toc154054299)

[2.1.5 Quản lý dữ liệu liên tục và Online learning: 32](#_Toc154054300)

[2.1.6 Learning Curve: 33](#_Toc154054301)

[2.1.7 Continual Learning Model: 35](#_Toc154054302)

[2.1.8 Quy trình học liên tục Continual Learning Pipeline: 36](#_Toc154054303)

[2.1.9 Data Management Strategy: 38](#_Toc154054304)

[2.2 Test Production: 39](#_Toc154054305)

[2.2.1 Model Testing (Kiểm thử mô hình): 40](#_Toc154054306)

[2.2.2 Kiểm thử hiệu suất (Performance Testing): 41](#_Toc154054307)

[2.2.3 System Testing (kiểm thử hệ thống): 43](#_Toc154054308)

[2.2.4 Kiểm thử tích hợp (Integration Testing): 44](#_Toc154054309)

[2.2.5 Model Deployment (Triển khai mô hình): 46](#_Toc154054310)

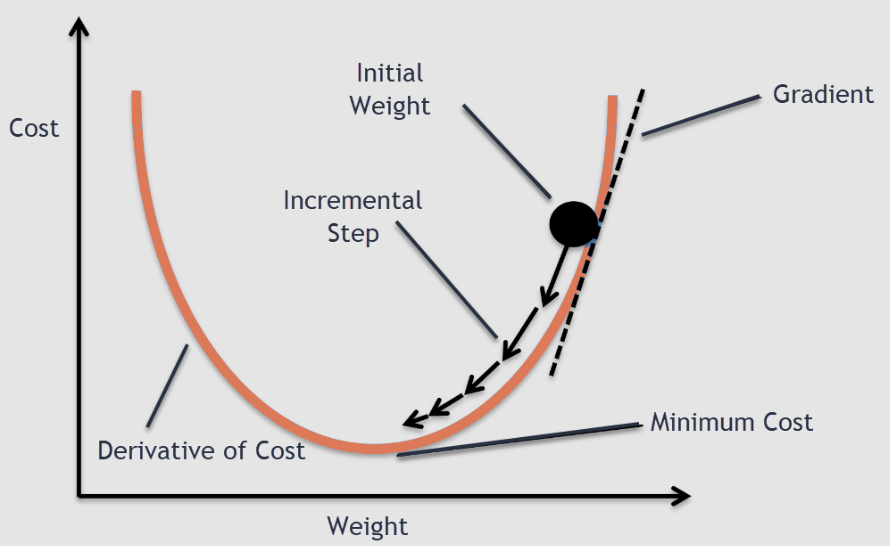
[2.2.6 Giám sản và quản lý mô hình (Model Monitoring and Management): 49](#_Toc154054311)

# Trình bày nghiên cứu và đánh giá về các vấn đề :

## Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình :

### Gradient Descent (GD):

* Đây là một thuật toán tìm tối ưu chung cho các hàm số. Ý tưởng chung của GD là điều chỉnh các tham số để lặp đi lặp lại thông qua mỗi dữ liệu huấn luyện để giảm thiểu hàm chi phí.



*Mô phỏng thuật toán Gradient Descent*

* Gradient Descent là một thuật toán tối ưu lặp (iterative optimization algorithm) được sử dụng trong các bài toán ML (Machine Learning ) và DL (Deep Learning) thường là các bà toán tối ưu lồi – Convex Optimization với mục tiêu là tìm một tập hợp các biến nội tại (internal parameters) cho việc tối ưu models. Trong đó:

1. Gradient: là tỷ lệ nghiêng của đường dốc (rate of inclination of a slope). Về mặt toán học, Gradient của một hàm là đạo hàm của hàm đó tương ứng với mỗi biến của hàm. Đối với hàm số đơn biến, chúng ta sử dụng khái niệm Derivative thay cho Gradient.
2. Descent: là từ viết tắt của Descending, và nghĩa là giảm dần.

* Gradient Descent có nhiều dạng khác nhau như Stochastic Gradient Descent, Mini-batch SDG. Nhưng về cơ bản thì đều được thực thi như sau:

1. Khởi tạo biến nội tại
2. Đánh giá model dựa vào biến nội tại và hàm mất mát (Loss Function)
3. Cập nhật các biến nội tại theo hướng tối ưu hàm mất mát (finding optimal points)
4. Lặp lại bước 2,3 cho tới khi thỏa điều kiện dừng.
5. Công thức cập nhật cho GD có thể được viết là:

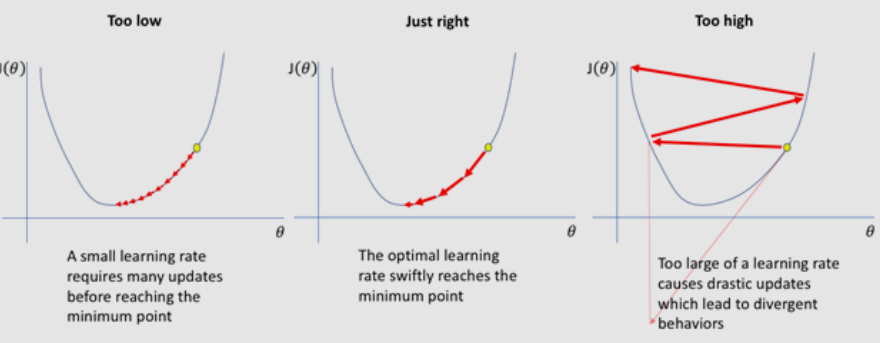
***Θ* = *θ* – *α* ⋅ ∇*J*(*θ*)**

+ ***θ***: là tập hợp các tham số mà chúng ta đang cố gắng tối ưu hóa.

+ ***α***: là **learning rate**, là một siêu tham số quyết định tốc độ học của thuật toán.

+ **∇*J*(*θ*)**: là gradient của hàm mất mát ***J*(*θ*)** đối với các tham số ***θ***. Gradient này được tính bằng cách lấy đạo hàm riêng của ***J*(*θ*)** đối với từng tham số.

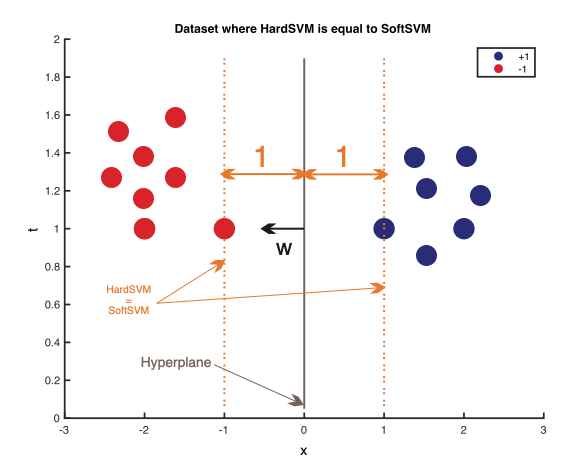
* Learning Rate: Đây là một tham số quan trọng trong DG, là giá trị của độ lớn của mỗi lần di chuyển (giống như độ dài sải chân khi leo xuống dốc). Tham số này được gọi là Learning Rate (tốc độ học). Nếu LR quá nhỏ, thuật toán sẽ phải thực hiện nhiều bước để hội tựu và sẽ mất nhiều thời gian. Và tuy nhiên nếu LR quá lớn thì sẽ khiến thuật toán đi qua cực tiểu, và vượt hẳn ra ngoài khiến thuật toán không thể hội tụ được.



***Sự ảnh hưởng của Learning Rate đến model***

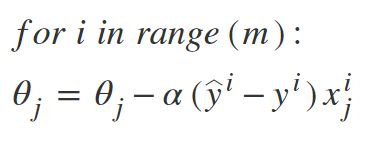
### Stochastic Gradient Descent (SGD):

* Trong thuật toán này tại một thời điểm, ta chỉ tính đạo hàm của hàm mất mát dựa trên chỉ một điểm dữ liệu xi rồi cập nhật θ dựa trên đạo hàm này. Việc này được thực hiện với từng điểm trên toàn bộ dữ liệu, sau đó lặp lại quá trình trên. Thuật toán rất đơn giải này trên thực tế lại làm việc rất hiệu quả.
* Mỗi lần duyệt một lượt qua tất cả các điểm trên toàn bộ dữ liệu được gọi là epoch. Với GD thông thường thì mỗi epoch ứng với 1 lần cập nhật θ, với SGD thì mỗi epoch ứng với N lần cập nhật θ với N điểm dữ liệu. Nhìn vào một mặt, việc cập nhật từng điểm một như thế này có thể làm giảm đi tốc độ thực hiện 1 epoch. Nhưng nhìn vào một mặt khác, SGD chỉ yêu cầu một lượng epoch rất nhỏ (thường là 10 cho lần đầu tiên, sau đó khi có dữ liệu mới thì chỉ cần chạy dưới một epoch là đã có nghiệm tốt). Vì vật SGD phù hợp với các bài toán có lượng cơ sở dữ liệu lớn (chủ yếu là Deep Learning).



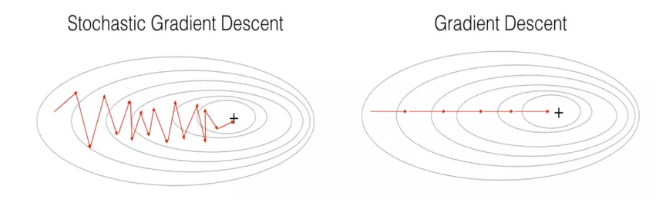
* Thứ tự lựa chọn điểm dữ liệu: một cách toán học, quy tắc cập nhật của SGD :

1. Một điểm cần lưu ý đó là: sau mỗi epoch, chúng ta cần cần xáo trộn thứ tự cảu các dữ liệu để đảm bảo tính ngẫu nhiên. Việc này cũng ảnh hưởng tới hiệu năng của SGD.
2. Một cách toán học, quy tắc cập nhật của SGD là:



Trong đó J(θ;xi;yi) là một hàm mất mát với chỉ 1 cặp điểm dữ liệu (input, label) là (Xi, Yi). Chú ý: chúng ta hoàn toàn có thể áp dụng các thuật toán tăng tốc GD như Momentum, Adagrad,... vào SGD.

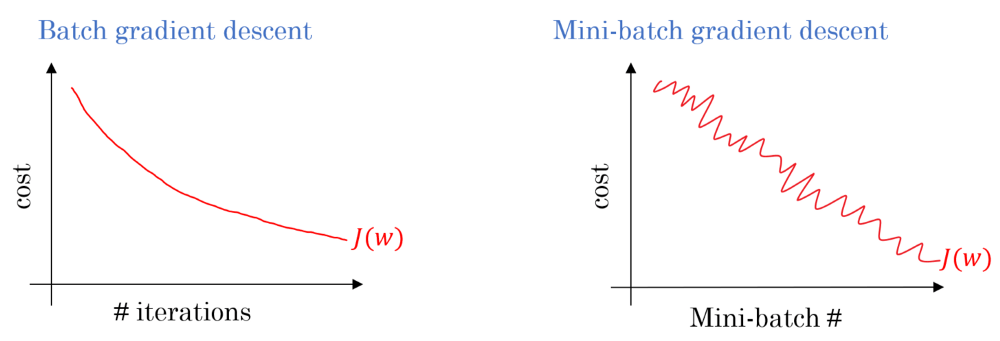
* Trong SGD vì chỉ có một cặp mẫu từ tập dữ liệu được chọn ngẫu nhiên cho mỗi lần lặp, nên đường dẫn mà thuật toán thực hiện để đến cực tiểu thường ồn ào hơn so với thuật toán Gradient Descent điển hình. Nhưng điều đó không quan trọng lắm vì đường đi của thuật toán không quan trọng, miễn là chúng ta đạt đến cực tiểu thời gian đào tạo gắn hơn đáng kể.



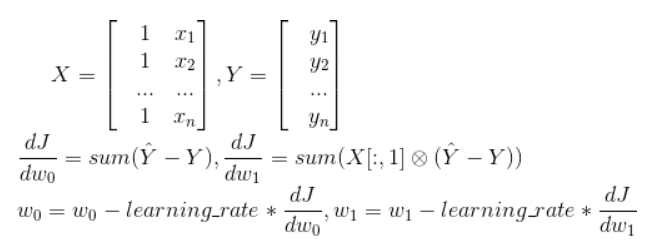
* Thực tế cho thấy chỉ lấy khoảng 10 điểm là ta đã có thể xác định được gần đúng phương trình đường thằng cần tìm rồi. Đây chính là ưu điểm của SGD – hội tụ rất nhanh.

### Mini-batch Gradient Descent:

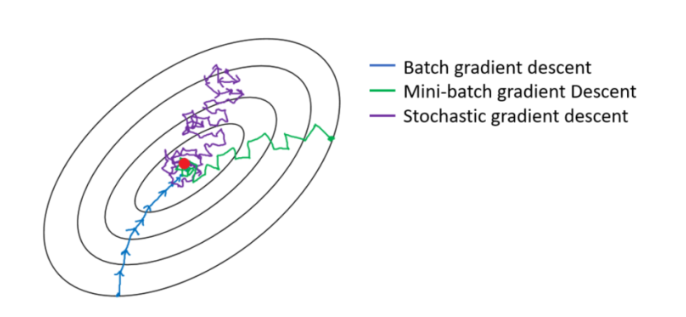
* Với Mini-BGD, traning dataset sẽ được chia thành nhiều gói (batch) khác nhau để xử lý. Sau khi tất cả cả dữ liệu trong batch, cost sẽ được tính toán và model sẽ được cập nhật tương ứng.
* Với cách tiếp cận này, tần suất cập nhật của model sẽ không quá nhiều như SGD và lượng dữ liệu cần phải xử lý trong một lần cũng nhỏ hơn SGD. Với khả năng cân bằng cùng việc cho deep learning. Tuy nhiên, chúng ta cần lưu ý rằng việc tùy chỉnh batch-size sẽ ảnh hưởng trực tiếp đến kết quả.
* Batch-size nhỏ giúp model học (converge) nhanh hơn nhưng dễ phát sinh nhiễu và thiếu ổn định.
* Batch-size lớn khiến model học (converge) chậm hơn nhưng sẽ ổn định và training chính xác hơn.



* Và ở trong thuật toán này, tại bước thứ hai khi ta tính đạo hàm của hàm loss và các biến. Trong bài linear regression ta dùng tất cả các dữ liệu trong dataset để tính đạo hàm rồi cập nhật bước 2:



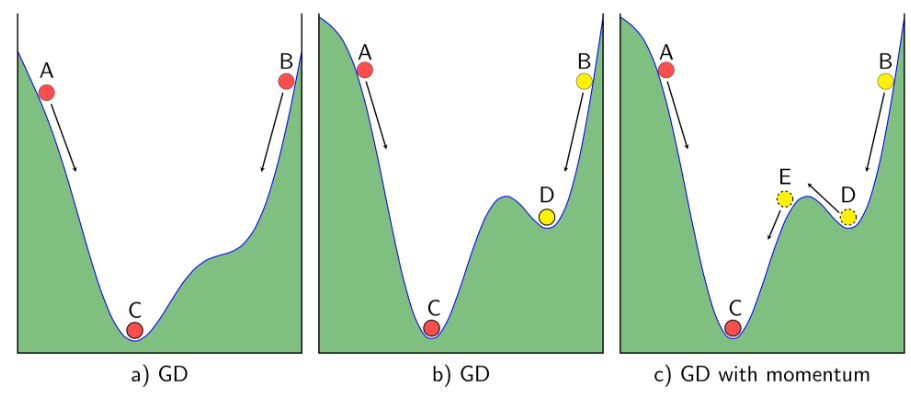
* Thuật toán gradient descent chạy tốt nhưng số lượng dữ liệu trong training set chỉ là 30. Tuy nhiên nếu dữ liệu có kích thước lớn như ảnh và số lượng lớn hơn ví dụ 5000 thì việc tính đạo hàm với hàm loss với toàn bộ dữ liệu sẽ rất tốn thời gian. Và mini-batch ra đời để giải quyết vấn đề đấy.



* Batch gradient descent thường được dùng khi số lượng dữ liệu trong traning set nhỏ hơn 2000. Với lượng dữ liệu lớn thì mini-batch gradient descent được sử dụng. Nó có thể giải quyết được vấn đề lượng dữ liệu quá lớn như trong batch gradient descent; hơn nữa đỡ nhiễu và có thể dùng vectorization so với stochastic gradient descent nên thường được sử dụng trong deep learning.

### Momentum:

* Nhắc lại thuật toán Gradient Descent : Để giải bài toán tìm điểm global optimal của loss function J(θ) với θ là tập hợp các tham số của model.
* Dưới góc nhìn vật lý: Thuật toán GD thường được ví với tác dụng của trọng lực lên một hòn bi đặt trên một mặt có dạng như hình một thung lũng giống như hình 1a) dưới đây. Bất kể ta đặt hòn bi ở A hay B thì cuối cùng hòn bi cũng sẽ lăn xuống và kết thúc ở vị trí C.



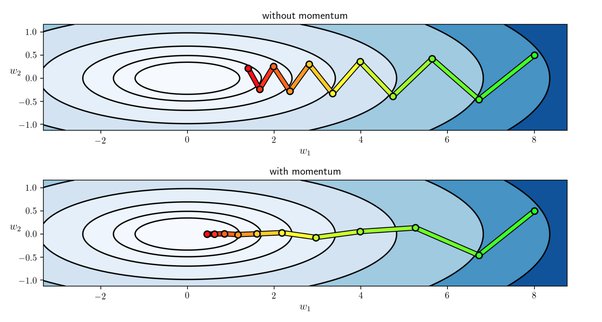
* Tuy nhiên, nếu như bề mặt có hay đáy thung lũng như hình 1b thì tùy vào việc đặt bi ở A hay B, vị trí cuối cùng của bi sẽ ở C hoặc D. Điểm D là một điểm local minimum chúng ta không mong muốn.
* Nếu suy nghĩ một cách vật lý hơn, vẫn trong Hình 1b), nếu vận tốc ban đầu của bi khi ở điểm B đủ lớn, khi bi lăn đến điểm D, theo đà, bi có thể tiếp tục di chuyển lên dốc phía bên trái của D. Và nếu giả sử vận tốc ban đầu lớn hơn nữa, bi có thể vượt dốc tới điểm E rồi lăn xuống C như trong Hình 1c). Đây chính là điều chúng ta mong muốn. Bạn đọc có thể đặt câu hỏi rằng liệu bi lăn từ A tới C có theo đà lăn tới E rồi tới D không.
* *Dựa trên hiện tượng này, một thuật toán được ra đời nhằm khắc phục việc nghiệm của GD rơi vào một điểm local minimum không mong muốn. Thuật toán đó có tên là Momentum (tức*theo đà*trong tiếng Việt).*
* **Gradient Descent với Momentum:** Trong GD, chúng ta cần tính lượng thay đổi ở thời điểm t để cập nhật vị trí mới cho nghiệm (tức hòn bi). Nếu chúng ta coi đại lượng này như vận tốc vt trong vật lý, vị trí mới của hòn bi sẽ là θt+1=θt−vt Dấu trừ thể hiện việc phải di chuyển ngược với đạo hàm. Công việc của chúng ta bây giờ là tính đại lượng vt sao cho nó vừa mang thông tin của độ dốc (tức đạo hàm), vừa mang thông tin của đà, tức vận tốc trước đó vt−1 (chúng ta coi như vận tốc ban đầu v0=0). Một cách đơn giản nhất, ta có thể cộng (có trọng số) hai đại lượng này lại:

**vt=γvt−1+η∇θJ(θ)**

* Trong đó γ thường được chọn là một giá trị khoảng 0.9, vt là vận tốc tại thời điểm trước đó, ∇θJ(θ) chính là độ dốc của điểm trước đó. Sau đó vị trí mới của hòn bi được xác định như sau:

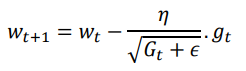
**θ=θ−vt**

* Thuật toán đơn giản này tỏ ra rất hiệu quả trong các bài toán thực tế (trong không gian nhiều chiều, cách tính toán cũng hoàn tòan tương tự). Dưới đây là một ví dụ trong không gian một chiều.



### Adagrad:

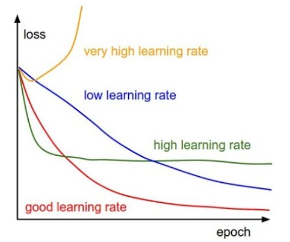
* Adagrad là một kỹ thuật học máy tiên tiến, thực hiện giảm dần độ dốc bằng cách thay đổi tốc độ học tập. Adagrad được cải thiện hơn bằng cách cho trọng số học tập chính xác dựa vào đầu vào trước nó để tự điều chỉnh tỉ lệ học theo hướng tối ưu nhất thay vì với một tỉ lệ học duy nhất cho tất cả các nút.



Trong đó :

1. *wt*,*i*​ là giá trị tham số tại thời điểm *t* và chiều thứ *i*.
2. *η* là tỷ lệ học (learning rate).
3. *gt*,*i*​ là gradient tại thời điểm *t* và chiều thứ *i*.
4. *Gt*,*ii*​ là tổng các bình phương gradient theo chiều thứ *i* cho tất cả các bước thời gian trước đó.
5. *ϵ* là một giá trị nhỏ để tránh chia cho 0.

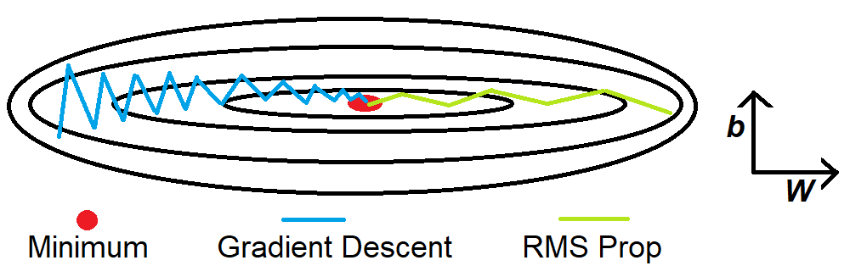
* Cụ thể, Adagrad giữ một bảng tính của bình phương của các gradient đã tích lũy cho mỗi tham số. Khi cập nhật trọng số, nó chia tỷ lệ học cho căn bậc hai của tổng các bình phương gradient đã tích lũy. Điều này giúp tỷ lệ học tự động thích ứng với mỗi tham số, giảm tỷ lệ học cho các tham số có gradient lớn và tăng tỷ lệ học cho các tham số có gradient nhỏ.
* Adagrad có thể giúp cải thiện hiệu suất của thuật toán tối ưu hóa trong trường hợp các tham số có gradient khác nhau đáng kể. Tuy nhiên, nó có một số nhược điểm, như việc giảm tỷ lệ học quá nhanh đối với các tham số có gradient lớn, có thể dẫn đến vấn đề vanishing learning rates. Do đó, có các biến thể của Adagrad được phát triển để khắc phục nhược điểm này.

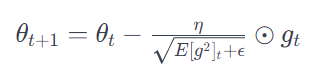


* **Ưu điểm**: Adagrad thích ứng tốt với dữ liệu có tham số có gradient biến động lớn và khác biệt.
* **Nhược điểm**: Mặc dù Adagrad có ưu điểm tự động điều chỉnh tỷ lệ học, nhưng nó có thể dẫn đến vấn đề vanishing learning rates. Điều này xảy ra khi bình phương của gradient tích lũy trở nên quá lớn, làm cho mẫu tiếp theo có ảnh hưởng quá nhỏ đến tỷ lệ học.
* **Regularization**: Adagrad tự nhiên có một tính chất của việc áp dụng L2 regularization (chuẩn hóa L2) vì nó thêm bình phương của gradient vào tổng.
* **Giá trị:** Để tránh việc chia cho 0 khi có gradient rất nhỏ, một giá trị nhỏ ϵ được thêm vào trong mẫu số của công thức cập nhật.

### RMSprop:

* RMSprop (Root Mean Square Propagation) là một thuật toán tối ưu hóa được sử dụng trong máy học và học sâu để cập nhật trọng số của mô hình. Thuật toán này giúp kiểm soát việc điều chỉnh kích thước của các bước cập nhật trọng số dựa trên lịch sử của các đạo hàm trước đó.
* Cụ thể, RMSprop tính toán một trung bình có trọng số của bình phương của các đạo hàm gradient. Nó sử dụng giá trị này để điều chỉnh kích thước của bước cập nhật cho mỗi trọng số. Mục tiêu là giảm độ lớn của gradient cho các trọng số có đạo hàm lớn và tăng độ lớn cho các trọng số có đạo hàm nhỏ. Điều này giúp giảm tốc độ học cho các trọng số có gradient lớn và tăng tốc độ học cho các trọng số có gradient nhỏ, giúp mô hình hội tụ nhanh chóng hơn.



* Công thức cập nhật trọng số của RMSprop được tính như sau:
* Trong đó:

1. *θt*+1​ là giá trị cập nhật mới của trọng số.
2. *θt*​ là giá trị trọng số hiện tại.
3. *gt*​ là gradient của hàm mất mát tại thời điểm *t*.
4. *E*[*g*2]*t*​ là trung bình có trọng số của bình phương của gradient cho mỗi trọng số.
5. *η* là tốc độ học (learning rate).
6. *ϵ* là một số nhỏ được thêm vào trong mẫu để tránh chia cho 0.

* Chi tiết hơn về RMSprop, ta có thể mô tả chi tiết các bước trong công thức cập nhật:

1. **Gradient Calculation**: Tại mỗi bước thời gian t, tính gradient gt của loss function theo các tham số *θ.*
2. **Squared Gradient** **Averaging** (Trung bình bình phương của Gradient): Tính toán bình phương của gradient cho mỗi trọng số *E*[*g*2]*t*​=*βE*[*g*2]*t*−1​+(1−*β*)*gt*2​. Trong đó *β* là hệ số giảm trọng số, thường được chọn là một giá trị nhỏ, ví dụ 0.9.
3. **Weight Update** (Cập nhật trọng số):

+ Cập nhật trọng số theo công thức:

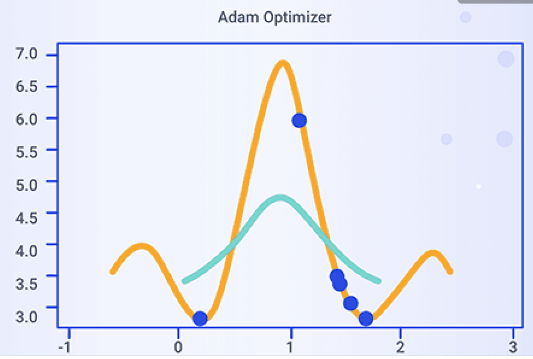
+ *η* Là tốc độ học (learning rate).

+ *ϵ* là một số nhỏ để tránh việc chia nhỏ 0.

* Thuật toán này giúp kiểm soát việc cập nhật trọng số bằng cách điều chỉnh kích thước của bước cập nhật dựa trên lịch sử gần đây của gradient. Nếu gradient đang biến động lớn, RMSprop sẽ giảm kích thước của bước cập nhật để giảm nguy cơ quá mức cập nhật. Ngược lại, nếu gradient ổn định, kích thước của bước cập nhật có thể tăng, giúp tăng tốc quá trình học.
* RMSprop giúp ổn định quá trình tối ưu hóa và thường được sử dụng trong các mô hình học sâu để đạt được hiệu suất tốt hơn, đặc biệt là khi các đặc trưng của dữ liệu thay đổi nhanh chóng và không đồng đều.

### Adam:

**Adam optimization**: à một phương pháp tối ưu hóa gradient-based được thiết kế để cải thiện hiệu suất của mô hình trong quá trình đào tạo. Dưới đây là một số chi tiết lý thuyết về Adam optimizer:



1. Gradient Descent Optimization:

* Adam thuộc loại tối ưu hóa gradient-based, nơi mục tiêu là điều chỉnh trọng số của mô hình dựa trên gradient của hàm mất mát theo trọng số đó.
* Các thuật toán tối ưu hóa gradient-based như Gradient Descent cố gắng tối thiểu hóa hàm mất mát bằng cách di chuyển theo hướng âm của gradient.

1. First Moment – Mean:

* Moment thứ nhất trong Adam tương ứng với trung bình có trọng số của gradient. Nó được tính bằng cách giữ lại một phần của gradient từ bước trước đó.
* Điều này giúp giảm độ biến động của gradient và làm cho quá trình đào tạo ổn định hơn.

1. Second Moment – Variance:

* Moment thứ hai trong Adam là trung bình trọng số của trung bình gradient. Nó giúp theo dõi sự kiện biến động của Gradient.
* Thông qua việc tích hợp moment thứ hai, Adam có thể thích ứng với tốc độ học tại từng tham số dựa trên mức biến động của gradient.\

1. Tính tỷ lệ học thích ứng (Adaptive learning Rate)

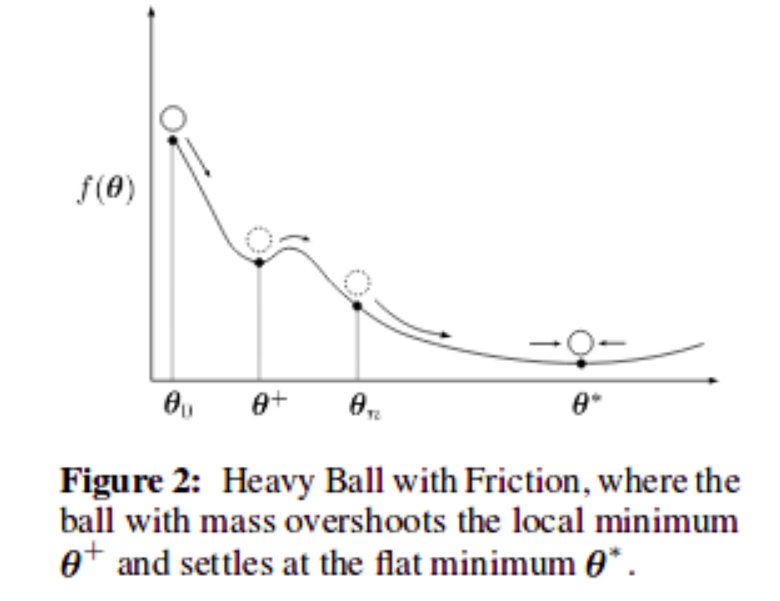
* Adam sử dụng tỷ lệ học thích ứng cho từng tham số để đảm bảo rằng mỗi tham số đều có một tỷ lệ học phù hợp
* Tỷ lệ học được điều chỉnh dựa trên moment thứ nhất và thứ hai, giúp đảm bảo học hiệu quả và tránh sự đánh mất thông tin quan trọng.

1. Bias-Corection:

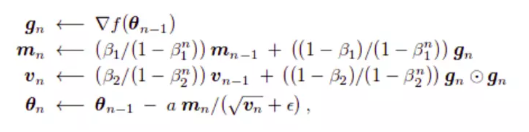
* Trong giai đoạn đầu của quá trình đảo, moment thứ nhất và thứ hai thể bị ảnh hưởng bởi giá trị khởi tạo ban đầu.
* Adam thực hiện một bước bias-correction để điều chỉnh ảnh hưởng của các moment, đảm bảo tính chính xác của chúng.

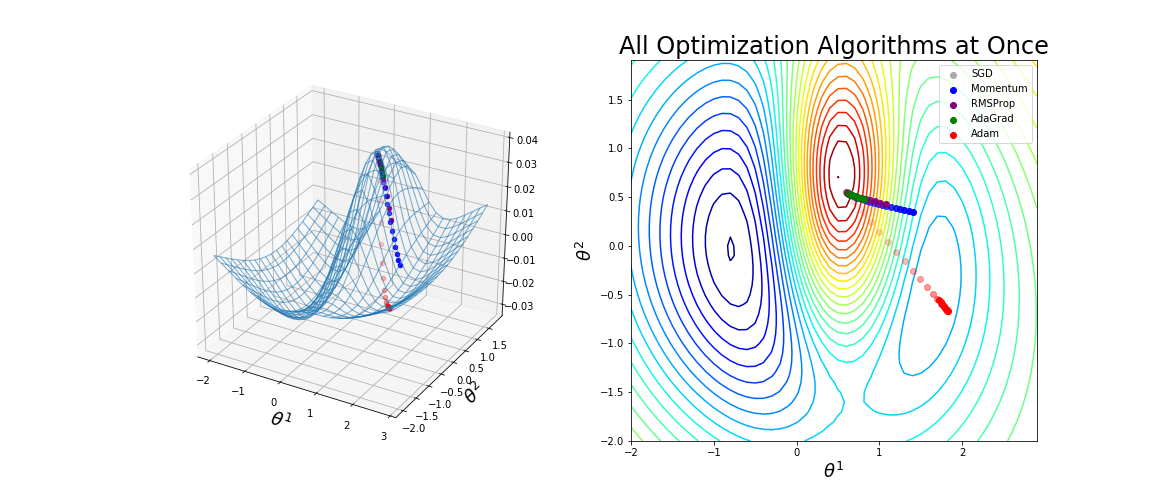
1. Độ giảm trọng số (Weight Decay):

* Adam hỗ trợ độ giảm trọng số bằng cách thêm một thành phần vào gradident để giảm giá trị của các trọng số.
* Điều này giúp kiểm soát việc mô hình trở nên quá phức tạp và ngăn chặn hiện tượng quá mức quá mức.
* Như đã nói ở trên Adam là sự kết hợp của Momentum và RMSprop . Nếu giải thích theo hiện tượng vật lí thì Momentum giống như 1 quả cầu lao xuống dốc, còn Adam như 1 quả cầu rất nặng có ma sát, vì vậy nó dễ dàng vượt qua local minimum tới global minimum và khi tới global minimum nó không mất nhiều thời gian dao động qua lại quanh đích vì nó có ma sát nên dễ dừng lại hơn.



* **Công thức :**

****

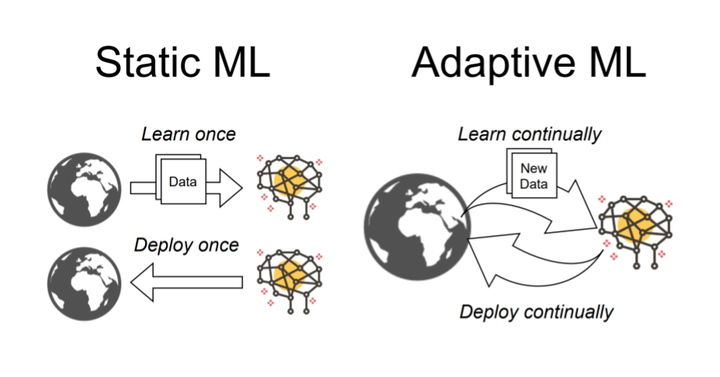


* Qua hình trên ta thấy optimizer 'Adam' hoạt động khá tốt, tiến nhanh tới mức tối thiểu hơn các phương pháp khác.
* **Các lợi ích của thuật toán Adam:**
* Không khó để implement.
* Độ phức tạp hiệu quả.
* Ít bộ nhớ yêu cầu.
* Thích hợp với các bài toán có độ biến thiên không ổn định và dữ liệu traning phân mảnh.
* Các siêu tham số được biến thiên một cách hiệu quả và yêu cầu ít điều chỉnh.

# Tìm hiểu về Continual Learning và Test Production:

## Continual Learning:

* Continual Learning, còn được biết đến với tên gọi Lifelong Learning, được xây dựng trên ý tưởng học liên tục về thế giới bên ngoài để khả năng phát triển tự động và gia tăng kiến thức và kỹ năng phức tạp hơn theo thời gian.

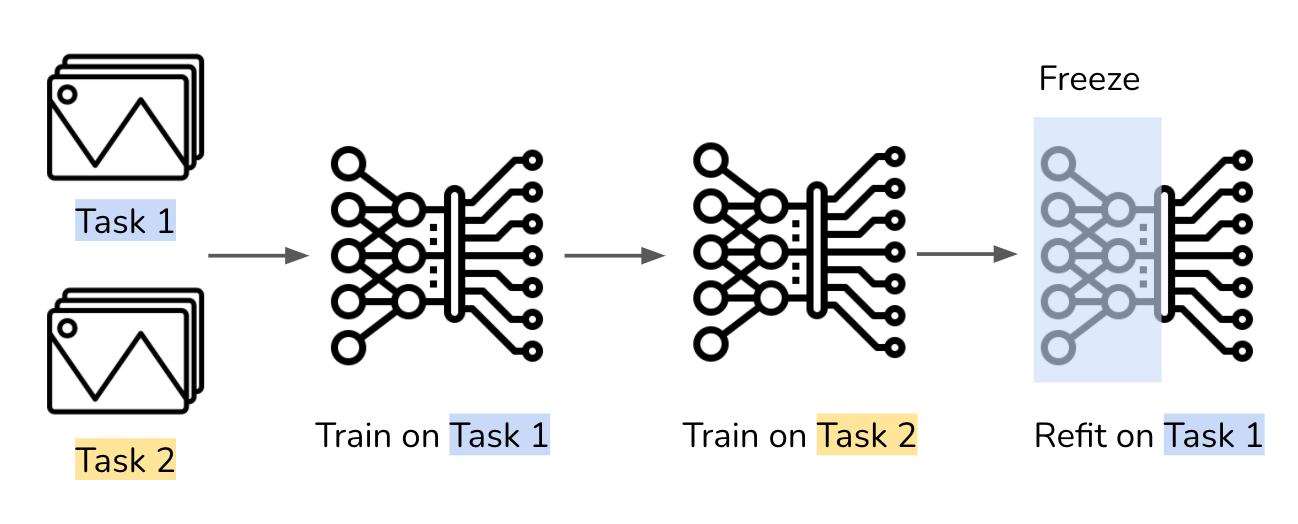


* Một hệ thống học liên tục có thể được định nghĩa là một thuật toán thích ứng có khả năng học từ một luồng thông tin liên tục, với thông tin đó trở nên ngày càng có sẵn theo thời gian và số lượng nhiệm vụ cần học (ví dụ: các lớp thành viên trong một nhiệm vụ phân loại) không được xác định trước. Quan trọng nhất, việc chấp nhận thông tin mới phải xảy ra mà không gây quên đối với kiến thức đã học trước đó hoặc gây ra xao lạc.
* Do đó, trong kịch bản Continual Learning, một mô hình học cần phải xây dựng một cách tăng dần và cập nhật động các biểu diễn nội tại khi phân phối của các nhiệm vụ thay đổi động theo thời gian. Lý tưởng, một phần của các biểu diễn nội tại này sẽ là tổng quát và không biến đổi đủ để có thể tái sử dụng cho các nhiệm vụ tương tự, trong khi một phần khác nên bảo toàn và mã hóa các biểu diễn cụ thể cho từng nhiệm vụ.

### Forgetting và Catastrophic Interference:

#### Định nghĩa:

* Catastrophic Forgetting, hay còn gọi là Nhiễu nhiễu Đột ngột, là một hiện tượng xảy ra khi một mạng neural hoặc mô hình học máy "quên" hoặc giảm đáng kể hiệu suất trên các nhiệm vụ đã học trước đó sau khi học một nhiệm vụ mới. Điều này có thể xảy ra khi đào tạo một mô hình trên một chuỗi các nhiệm vụ thay vì đào tạo nó trên tất cả các nhiệm vụ cùng một lúc.

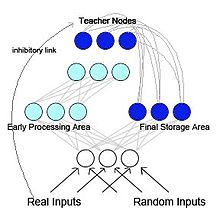


#### Nguyên nhân của Forgetting:

* **Overfitting**: Khi mô hình tập trung quá mức vào việc "vừa vặn" dữ liệu đào tạo của nhiệm vụ mới, có thể dẫn đến quá mức thích ứng với các đặc điểm cụ thể của nhiệm vụ đó, và do đó, mô hình có thể mất khả năng tổng quát và quên thông tin của các nhiệm vụ trước đó.
* **Interference**: Khi một nhiệm vụ mới liên quan đến các nhiệm vụ trước đó, quá trình học về nhiệm vụ mới có thể tác động tiêu cực đến kiến thức của mô hình về các nhiệm vụ trước đó. Sự can thiệp này gọi là nhiễu nhiễu và làm suy giảm hiệu suất của mô hình trên các nhiệm vụ trước đó.
* **Online Learning**: Trong phương pháp học trực tuyến, mô hình được cập nhật liên tục với dữ liệu mới khi nó xuất hiện. Nếu dữ liệu mới khác biệt đáng kể so với dữ liệu đã được sử dụng trước đó, mô hình có thể không giữ được kiến thức từ quá khứ và dễ gặp hiện tượng quên.

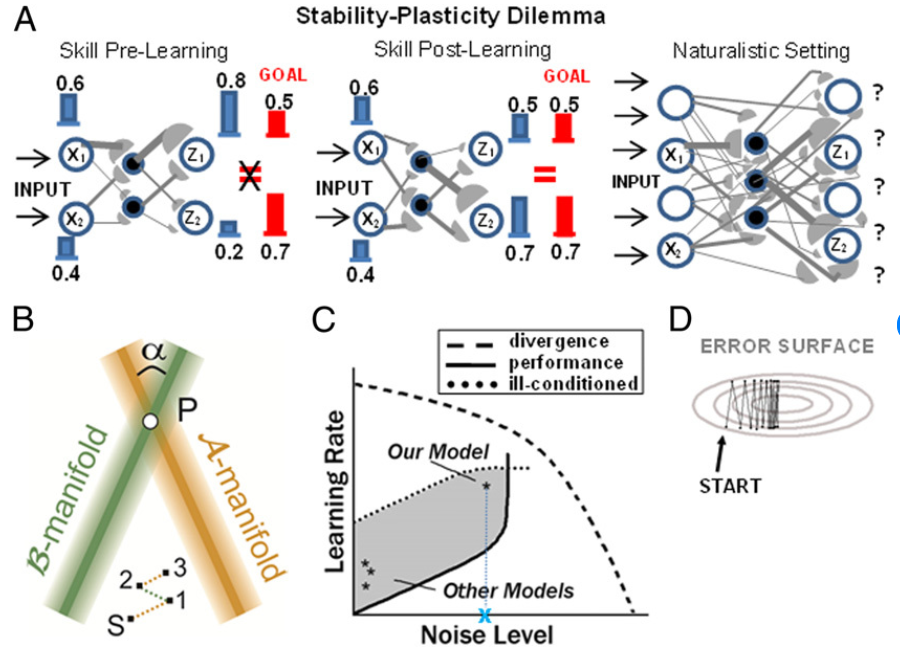
#### Nguyên nhân của Catastrophic Interference:

* **Tính liên quan của các nhiệm vụ**: khi các nhiệm vụ liên quán đến nhau, sự học về một nhiệm vụ mới có thể làm thay đổi trọng số của mô hình một cách đáng kể và làm ảnh hưởng đến khả năng của nó trong việc giữ lại thông tin của các nhiệm vụ trước đó.
* **Thay đổi trọng số quá lớn**: quá trình học về một nhiệm vụ mới có thể dẫn đến thay đổi trọng số quá lớn, đặc biệt là nếu các nhiệm vụ liên quán đến nhau. Điều này có thể gây ra hiện tượng can thiệp đột ngột và làm suy giảm hiệu suất trên các nhiệm vụ trước đó.
* **Không gian trạng thái hạn chế**: nếu không gian trạng thái của mô hình hạn chế, mô hình có thể không đủ linh hoạt để duy trì kiến thức từ nhiều nhiệm vụ mà không làm suy giảm hiệu suất.
* Để giải quyết các vấn đề này, các phương pháp như "Elastic Weight Consolidation" (EWC), "Rehearsal", và "Transfer Learning" đã được phát triển để giảm thiểu hiện tượng Forgetting và Catastrophic Interference trong quá trình học máy.



### Plasticity và Stability:

* "Plasticity" và "Stability" là hai khái niệm quan trọng trong lĩnh vực Học máy và Neuroplasticity (năng động học thần kinh). Dưới đây là giải thích chi tiết về cả hai khái niệm:



#### Plasticity:

* **Định nghĩa**: Plasticity đề cập đến khả năng thay đổi cấu trúc và chức năng của mạng neural hoặc hệ thống học máy dựa trên kinh nghiệm. Nó có nghĩa là mô hình có khả năng thích ứng và học từ dữ liệu mới, điều này giúp nó nhanh chóng thích nghi với môi trường và tác vụ mới.
* **Ví dụ trong não học**: Năng động thần kinh (neuroplasticity) là khả năng của não để thích ứng và thay đổi kết nối giữa các tế bào thần kinh dựa trên kinh nghiệm và học hỏi. Điều này đặc biệt quan trọng trong quá trình học và phát triển.
* **Trong học máy**: Mô hình có năng động cao thường có khả năng học nhanh chóng và thích ứng tốt với dữ liệu mới, nhưng cũng có thể dễ dàng bị ảnh hưởng bởi hiện tượng quên nếu không được kiểm soát.

#### Stability:

* **Định nghĩa**: Stability là khả năng của mạng neural hoặc hệ thống học máy duy trì hiệu suất và thông tin đã học mà không bị ảnh hưởng quá mức bởi dữ liệu mới hoặc các thay đổi trong môi trường.
* **Ví dụ trong não học**: Một não ổn định có khả năng duy trì kiến thức và kỹ năng đã học mà không bị mất mát quá nhiều khi đối mặt với thông tin mới.
* **Trong học máy**: Mô hình ổn định giúp ngăn chặn hiện tượng quên và nhiễu nhiễu đột ngột. Nó có thể duy trì hiệu suất trên các nhiệm vụ trước đó khi học nhiệm vụ mới mà không làm suy giảm khả năng tổng quát của mô hình.

#### Mối quan hệ giữa Plasticity và Stability:

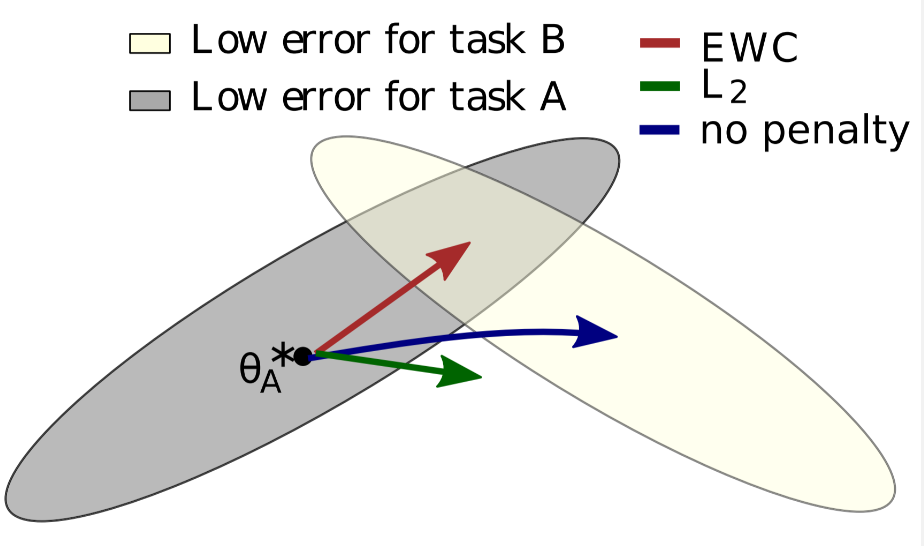
#### Sự cân bằng giữa Plasticity và Stability

* **Plasticity Cao và Stability Thấp**:
* *Ưu điểm*: Mô hình có khả năng nhanh chóng học từ dữ liệu mới và thích ứng với môi trường mới.
* *Nhược điểm*: Có thể dễ bị ảnh hưởng bởi hiện tượng quên, nơi mà thông tin từ nhiệm vụ trước đó có thể bị mất khi học nhiệm vụ mới.
* **Plasticity Thấp và Stability Cao:**
* Ưu điểm: Mô hình duy trì ổn định và giữ lại thông tin từ các nhiệm vụ trước đó tốt.
* Nhược điểm: Có thể hạn chế khả năng học từ dữ liệu mới và thích ứng với môi trường đổi biến.

#### Các phương pháp và kỹ thuật can thiệp

#### Elastic weight consolidation (EWC):

* Mục tiêu: Giữ lại trọng số quan trọng liên quan đến nhiệm vụ trước đó, ngăn chặn sự thay đổi quá mức khi học nhiệm vụ mới.
* Tác động: Tăng sự ổn định của mô hình bằng cách giữ lại thông tin quan trọng từ quá khứ.



#### Rehearsal (Luyện tập lại):

* Mục tiêu: Duy trì kiến thức từ các nhiệm vụ trước đó bằng cách luyện tập lại mô hình với dữ liệu cũ.
* Tác động: Tăng cường plasticity bằng cách liên tục đưa thông tin cũ vào quá trình học.

#### Transfer Learning (học chuyển giao):

* Mục tiêu: Chia sẻ kiến thức từ một nhiệm vụ cho một nhiệm vụ liên quan.
* Tác động: Tăng cường khả năng học từ dữ liệu mới bằng cách sử dụng kiến thức đã học từ các nhiệm vụ trước đó.

#### Biểu hiện trong hệ thống thần kinh thực tế:

* Trong não người, sự cân bằng giữa năng động và ổn định thường được duy trì thông qua cơ chế như synaptic pruning (cắt giảm liên kết thần kinh không cần thiết) và long-term potentiation (tăng cường lâu dài), giúp kiểm soát việc thay đổi trong cấu trúc và chức năng thần kinh.

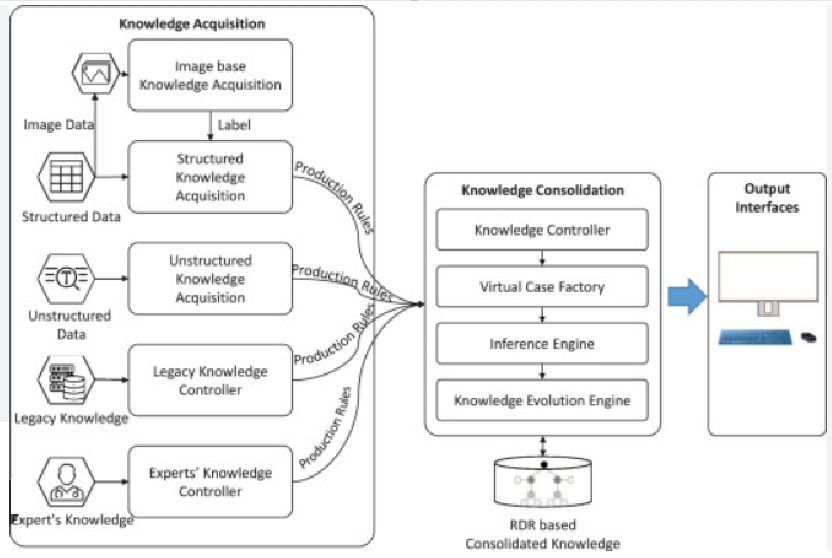
#### Thách thức và nghiên cứu tiếp theo:

* Cân bằng giữa Plasticity và Stability là một thách thức đối với các nhà nghiên cứu và nhà phát triển mô hình học máy. Nghiên cứu tiếp theo liên quan đến cách tối ưu hóa cân bằng này là quan trọng để tạo ra các mô hình có khả năng học mạnh mẽ và ổn định trong các tình huống thực tế.

### Knowledge Consolidation:

#### Định nghĩa:

* Knowledge Consolidation (Tổng hợp kiến thức) trong machine learning đề cập đến quá trình hệ thống hóa và tăng cường thông tin mà mô hình đã học từ dữ liệu, đặc biệt là khi mô hình phải đối mặt với nhiều nhiệm vụ và dữ liệu mới.
* Mục tiêu là duy trì và cải thiện hiệu suất của mô hình trên các nhiệm vụ trước đó mà nó đã học, ngăn chặn hiện tượng quên đột ngột và giữ lại kiến thức đã đạt được.



#### Vai trò và ý nghĩa:

* **Đuối trọng dữ liệu cũ**: Knowledge Consolidation giúp mô hình giữ lại thông tin quan trọng từ dữ liệu cũ, nhằm giảm thiểu tác động của việc học nhiệm vụ mới lên kiến thức cũ.
* **Đối phó với Catastrophic Forgetting**: Quá trình Knowledge Consolidation được thiết kế để đối phó với hiện tượng Catastrophic Forgetting, nơi mô hình quên một cách đột ngột kiến thức từ nhiệm vụ trước khi học nhiệm vụ mới.
* **Học đa nhiệm và học liên tục**: Knowledge Consolidation hỗ trợ mô hình trong việc học đa nhiệm (multi-task learning) và học liên tục (continual learning) mà không làm suy giảm hiệu suất trên các nhiệm vụ đã học trước đó.

#### Phương pháp và chiến lược:

* **Elastic Weight Consolidation (EWC)**: Một phương pháp phổ biến, EWC giữ lại trọng số quan trọng của mô hình liên quan đến nhiệm vụ trước đó, ngăn chặn sự thay đổi quá mức khi học nhiệm vụ mới.
* **Rehearsal**: Mô hình được huấn luyện lại với các ví dụ từ dữ liệu cũ để củng cố kiến thức và thông tin quan trọng.
* **Transfer Learning**: Sử dụng kiến thức đã học từ một nhiệm vụ để hỗ trợ học một nhiệm vụ liên quan.

#### Thách thức và nghiên cứu tiếp theo:

* **Ứng dụng trong đời thực**: Knowledge Consolidation đang đối mặt với thách thức trong việc triển khai trong các hệ thống thực tế, đặc biệt là khi mô hình phải xử lý dữ liệu lớn và đa dạng.
* **Tiếp tục nghiên cứu**: Cần thêm nghiên cứu để phát triển phương pháp và chiến lược hiệu quả hơn trong quá trình Knowledge Consolidation, đặc biệt là trong bối cảnh học liên tục và đa nhiệm.

### Transfer Learning và Fine-tuning:

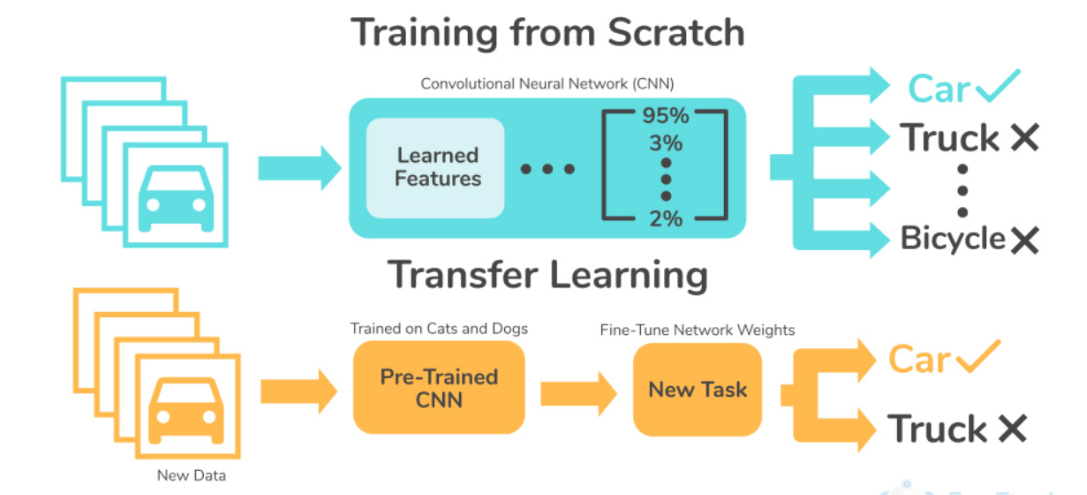
#### Transfer Learning trong machine learning:

#### Định nghĩa:

* Transfer Learning (Học Chuyển Giao) là một phương pháp trong machine learning mà mô hình được huấn luyện trước đó trên một tác vụ lớn có sẵn, sau đó được chuyển giao để thực hiện trên một tác vụ liên quan nhưng không giống hệt.

#### Ưu điểm:

* **Tiết kiệm tài nguyên**: Giảm đòi hỏi về dữ liệu và thời gian đào tạo bằng cách sử dụng kiến thức từ một tác vụ sang một tác vụ khác.
* **Hiệu suất cao hơn**: Thường dẫn đến hiệu suất cao hơn so với việc huấn luyện mô hình từ đầu, đặc biệt khi dữ liệu đào tạo cho tác vụ mới ít.



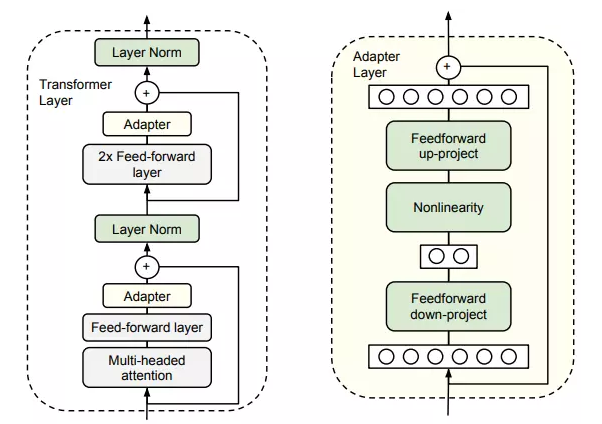
#### Phương pháp chuyển giao:

* **Feature Extraction**: Sử dụng các đặc trưng đã học từ mô hình trước đó như một bộ trích xuất đặc trưng cho tác vụ mới.
* **Fine-tuning**: Tiếp tục đào tạo mô hình trên tập dữ liệu mới để cập nhật các trọng số và thích ứng với tác vụ mới.

#### Fine-tuning trong Machine learning:

#### Định nghĩa

* Fine-tuning (Tinh Chỉnh) là quá trình điều chỉnh và cập nhật trọng số của mô hình đã được huấn luyện trước đó (thông thường từ Transfer Learning) trên tập dữ liệu mới của một tác vụ cụ thể.



#### Các bước thực hiện

* Chọn mô hình cơ sở (base model): Sử dụng một mô hình đã được huấn luyện trước đó trên một tập dữ liệu lớn.
* Dùng tính chỉnh: Tiếp tục huấn luyện mô hình trên tập dữ liệu mới của tác vụ cụ thể.
* Cập nhật trọng số: Cập nhật trọng số của mô hình để nó có khả năng tốt hơn đối với tập dữ liệu mới.

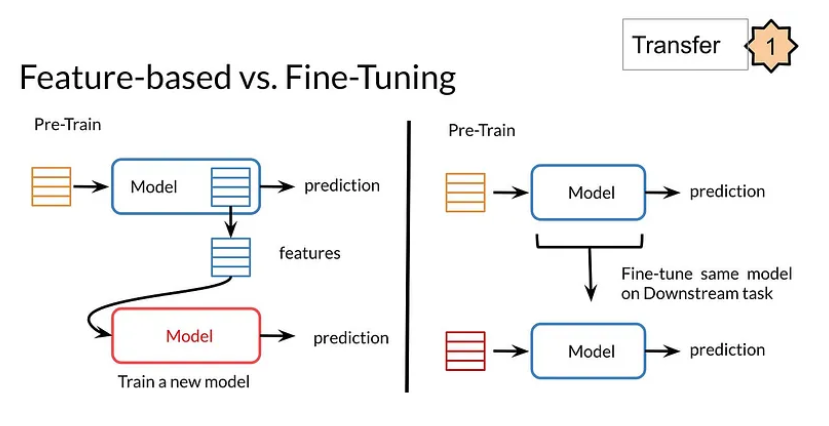
#### Ưu điểm:

* Hiệu suất tốt hơn: Tinh chỉnh giúp mô hình chuyển giao thích ứng chính xác hơn với tập dữ liệu cụ thể của tác vụ mới.
* Tiết kiệm thời gian: Thường không yêu cầu nhiều thời gian đào tạo so với việc huấn luyện mô hình từ đầu.

#### Thách thức:

* Quản lý overfitting: Cần quản lý rủi ro overfitting, đặc biệt nếu tập dữ liệu mới có kích thước nhỏ.

#### So sánh Transfer Learning và Fine-turning:



* **Transfer learning**: Tập trung vào việc chuyển giao kiến thức từ một tác vụ sang tác vụ khác.
* **Fine-tuning**: Là phần cuối của quá trình chuyển giao, nơi mô hình được điều chỉnh để phản ánh thông tin từ tập dữ liệu mới.
* **Tinh chỉnh là phần của chuyển giao**: Fine-tuning thường được thực hiện sau khi đã chuyển giao kiến thức từ một mô hình đã được huấn luyện trước đó.
* **Mục tiêu chính khác nhau**: Transfer Learning nhấn mạnh sự chia sẻ kiến thức, trong khi Fine-tuning tập trung vào việc điều chỉnh mô hình để phù hợp với tập dữ liệu mới.
* Cả hai phương pháp đều làm tăng hiệu suất của mô hình và giảm yêu cầu về dữ liệu và thời gian đào tạo.

### Quản lý dữ liệu liên tục và Online learning:

* Quản lý Dữ Liệu Liên Tục và Học Trực Tuyến là một khía cạnh quan trọng trong lĩnh vực Continual Learning, đặc biệt là khi mô hình cần thích ứng với dữ liệu mới một cách liên tục và đồng thời không làm suy giảm hiệu suất trên các nhiệm vụ đã học.

#### Quản lý dữ liệu liên tục:

* Trong môi trường Continual Learning, dữ liệu liên tục được đưa vào mô hình theo thời gian thay vì một lượng lớn dữ liệu tĩnh.
* Quản lý dữ liệu liên tục đòi hỏi mô hình có khả năng học từ mẫu dữ liệu mới mà không ảnh hưởng đến hiệu suất trên mẫu dữ liệu cũ.
* Các chiến lược như lưu trữ mẫu dữ liệu quan trọng, sử dụng kỹ thuật đánh trọng số cho dữ liệu mới, và sử dụng đồng thời dữ liệu cũ và mới có thể được áp dụng để giảm thiểu hiện tượng quên thảm họa.

#### Online-learning:

* Học trực tuyến là quá trình học mà mô hình được cập nhật ngay khi có dữ liệu mới, thay vì đợi đến khi có một lượng lớn dữ liệu mới để đào tạo lại toàn bộ mô hình.
* Việc học trực tuyến giúp mô hình linh hoạt và có khả năng thích ứng nhanh chóng với môi trường biến đổi.
* Điều này đặc biệt quan trọng trong các ứng dụng yêu cầu phản ứng nhanh đối với dữ liệu mới, ví dụ như trong hệ thống giám sát và dự đoán thời gian thực.

#### Thách thức của online learning:

* Mặc dù học trực tuyến mang lại sự linh hoạt, nhưng cũng đặt ra thách thức về tính ổn định của mô hình, vì nó cần phải thích ứng ngay lập tức với mỗi dữ liệu mới.
* Các kỹ thuật như giảm trọng số, học từ gradient mới, và kiểm soát tỉ lệ học có thể được sử dụng để duy trì ổn định và tránh quá trình học quá mức từ dữ liệu mới.

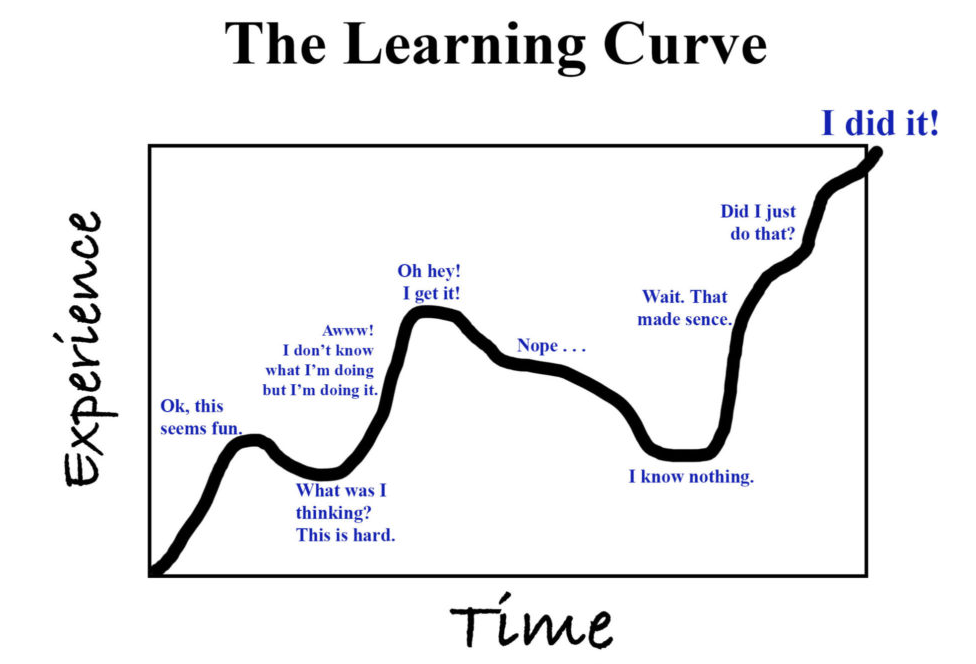
#### Incremental Continual Learning:

* Trong môi trường học liên tục tăng dần, mô hình được mở rộng khi có dữ liệu mới hoặc nhiệm vụ mới, giúp duy trì kiến thức cũ và học kiến thức mới một cách liên tục.
* Mô hình có thể được điều chỉnh và mở rộng dựa trên các nhiệm vụ cụ thể mà không làm ảnh hưởng đến các phần đã học trước đó.
* Quản lý Dữ Liệu Liên Tục và Học Trực Tuyến là yếu tố quan trọng để đảm bảo rằng mô hình có khả năng học liên tục mà không làm suy giảm hiệu suất trên dữ liệu cũ và có thể thích ứng linh hoạt với môi trường biến đổi. Nghiên cứ

### Learning Curve:

#### Hiểu rõ về quá trình học:

* Mỗi khi mô hình được đào tạo trên một tập dữ liệu mới, nó sẽ có một biểu đồ học sinh, thể hiện sự thay đổi trong hiệu suất của mô hình qua thời gian.
* Biểu đồ này giúp quan sát sự tiến triển của mô hình trong quá trình học và đánh giá khả năng học liên tục.



#### Nguyên tắc ứng dụng trên nhiều nhiệm vụ:

* Nguyên tắc này không chỉ áp dụng cho một nhiệm vụ duy nhất mà còn mở rộng sang nhiều nhiệm vụ khác nhau mà mô hình cần thực hiện.
* Việc theo dõi sự tiến triển qua thời gian giúp xác định liệu mô hình có thể học một cách liên tục và giữ lại kiến thức từ các nhiệm vụ cũ hay không.

#### Quản ánh quá trình học:

* Learning Curve phản ánh cách mô hình đáp ứng với dữ liệu mới. Nó có thể thể hiện liệu mô hình có xuất hiện hiện tượng quên thảm họa hay không, nghĩa là liệu hiệu suất trên nhiệm vụ trước đó có giảm đáng kể sau khi học từ dữ liệu mới hay không.

#### Quản lý độ khó:

* Learning Curve cũng có thể cung cấp thông tin về độ khó của nhiệm vụ. Nếu biểu đồ học sinh cho thấy sự giảm độ chính xác trên thời gian, có thể đây là dấu hiệu của việc mô hình gặp khó khăn trong việc học từ dữ liệu mới.

#### Điều chỉnh phương pháp học:

* Dựa trên biểu đồ học sinh, người ta có thể điều chỉnh phương pháp học của mô hình. Nếu có hiện tượng quên thảm họa, có thể xem xét việc sử dụng các kỹ thuật như regularization, transfer learning, hoặc thậm chí fine-tuning để cải thiện hiệu suất.

#### Đánh giá tiềm năng học liên tục:

* Learning Curve là một công cụ quan trọng để đánh giá tiềm năng của mô hình trong việc học liên tục và đảm bảo rằng nó có thể tự động thích ứng với dữ liệu mới mà không gây quên kiến thức cũ

.

### Continual Learning Model:

* Mô hình học liên tục (Continual Learning) là một phương pháp trong lĩnh vực máy học, nơi mô hình được đào tạo để liên tục học từ dữ liệu mới mà không quên kiến thức đã học trước đó. Điều này là quan trọng trong các tình huống mà dữ liệu thay đổi theo thời gian hoặc khi mô hình cần phải xử lý dữ liệu đến từ nhiều nguồn khác nhau.

#### Memory Replay (Tái tạo bộ nhớ):

* Một trong những phương pháp phổ biến trong học liên tục là lưu trữ một số lượng dữ liệu cũ và sử dụng chúng lại khi cần thiết để đào tạo lại mô hình. Điều này giúp mô hình giữ lại kiến thức cũ trong quá trình học từ dữ liệu mới.

#### Regularization Techniques:

* Sử dụng các kỹ thuật chính quy như Elastic Weight Consolidation (EWC) hoặc Synaptic Intelligence giúp giữ nguyên trọng số quan trọng của mô hình đối với dữ liệu cũ trong khi vẫn học từ dữ liệu mới

#### Dynamic Architechtures:

* Một số phương pháp liên quan đến việc thay đổi cấu trúc của mô hình dựa trên dữ liệu mới. Điều này có thể bao gồm việc thêm hoặc loại bỏ các lớp mạng, điều chỉnh số lượng nơ-ron, hoặc thậm chí tạo ra các mô-đun mở rộng.

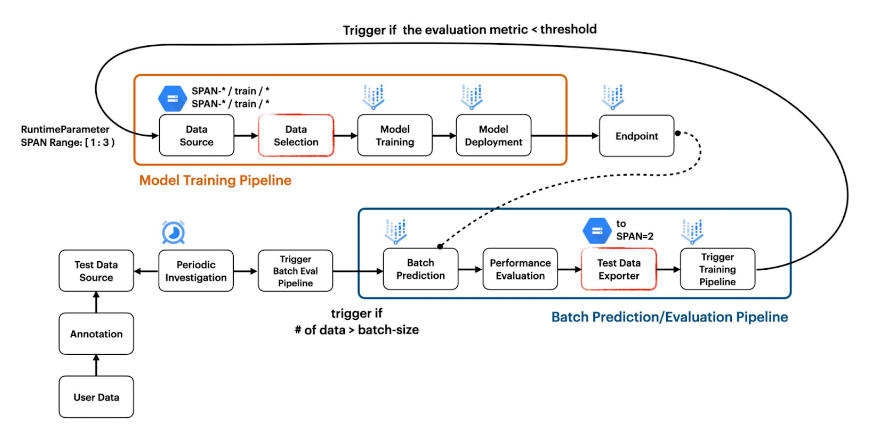
#### Gradient Episodic Memory:

* GEM là một phương pháp giúp giải quyết vấn đề quên khi đào tạo mô hình trên nhiều nhiệm vụ. Nó giữ lại thông tin quan trọng trong quá trình học mới mà không làm ảnh hưởng đến hiệu suất trên các nhiệm vụ đã học trước đó.

#### Task-Incremantal Learning:

* Thay vì đào tạo mô hình cho tất cả các nhiệm vụ cùng một lúc, ta có thể giới thiệu từng nhiệm vụ một cách tăng dần, giúp mô hình tập trung hơn vào từng nhiệm vụ cụ thể.

### Quy trình học liên tục Continual Learning Pipeline:



#### Chuẩn bị dữ liệu:

* Thu thập dữ liệu: Thu thập dữ liệu từ nguồn tương ứng với nhiệm vụ cụ thể.
* Chuẩn hóa dữ liệu: Chuẩn hóa và tiền xử lý dữ liệu để đảm bảo đồng nhất và đảm bảo rằng nó phù hợp với mô hình hiện tại.

#### Chọn mô hình ban đầu:

* Lựa chọn mô hình cơ bản: Chọn một mô hình học máy cơ bản để bắt đầu quá trình học.
* Khởi tạo trọng số: Khởi tạo trọng số của mô hình và chuẩn bị nó cho quá trình học.

#### Đào tạo mô hình ban đầu:

* Đào tạo ban đầu: Sử dụng dữ liệu hiện tại để đào tạo mô hình cơ bản.
* Đánh giá hiệu suất ban đầu: Đánh giá hiệu suất của mô hình trên dữ liệu kiểm định để đo lường khả năng tổng quát.

#### Học liên tục:

* Xác định nhiệm vụ mới: Xác định nhiệm vụ mới hoặc nhóm dữ liệu cần được học.
* Chuẩn bị dữ liệu mới: chuẩn bị và tiền xử lý dữ liệu để đưa vào quá trình học.
* Đào tạo với dữ liệu mới: Tiếp tục đào tạo mô hình với dữ liệu mới, có thể kết hợp với các dữ liệu cũ.

#### Kiểm thử và đánh giá:

* Đánh giá hiệu suất toàn diện: đánh giá hiệu suất của mô hình trên cả nhiệm vụ đã học trước đó và nhiệm vụ mới.
* Kiểm tra đồng nhất: kiểm tra xem mô hình có giữ lại khả năng tổng quát qua thời gian không.

#### Lặp lại quá trình:

* Lặp lại quá trình học liên tục: nếu có nhiều nhiệm vụ hoặc dữ liệu mới, lặp lại quá trình học liên tục.

#### Bảo dưỡng và cập nhật:

* Kiểm tra hiệu suất định kỳ: Thực hiện kiểm tra định kỳ để đảm bảo rằng mô hình vẫn hoạt hiệu quả trên tất cả các nhiệm vụ.
* Cập hình mô hình nếu cần: Nếu có dữ liệu mới hoặc nhiệm vụ mới, cập nhật mô hình để giữ lại và tích hợp kiến thức mới.

### Data Management Strategy:

* Chiến lược quản lý dữ liệu là cách mà hệ thống quản lý và sử dụng dữ liệu mới mà không gây ảnh hưởng lớn đến hiệu suất trên dữ liệu cũ.
* Các chiến lược này bao gồm lưu trữ dữ liệu quan trọng, đánh trọng số cho dữ liệu mới, sử dụng dữ liệu cũ để đào tạo lại mô hình, và các kỹ thuật khác.

#### Data Consistency:

* Đảm bảo sự đồng nhất và đồng đều trong dữ liệu từ các nguồn khác nhau là quan trọng để mô hình có thể học một cách hiệu quả. Điều này có thể đòi hỏi chuẩn hóa dữ liệu và xử lý ngoại lệ.

#### Big Data Management:

* Nếu bạn có một lượng lớn dữ liệu, chiến lược quản lý dữ liệu phải có khả năng xử lý và lưu trữ dữ liệu lớn một cách hiệu quả. Điều này có thể bao gồm việc sử dụng các hệ thống cơ sở dữ liệu phân tán và các kỹ thuật xử lý lớn.

#### Memory Modes:

* Xác định xem liệu dữ liệu có thể được giữ lại (memory replay) hay không. Nếu có, cần xác định kích thước của bộ nhớ để lưu trữ các mẫu quan trọng để đào tạo lại mô hình.

#### Training Cycles:

* Xác định bao lâu bạn sẽ đào tạo lại mô hình với dữ liệu mới. Điều này có thể phụ thuộc vào tần suất cập nhật dữ liệu và tốc độ thay đổi của nó.

#### Multitask Data Management:

* Trong trường hợp học liên tục với nhiều nhiệm vụ, chiến lược cần phải quản lý dữ liệu từ các nhiệm vụ khác nhau và đảm bảo rằng chúng không gây nhiễu loạn cho nhau.

#### Regularization During Training:

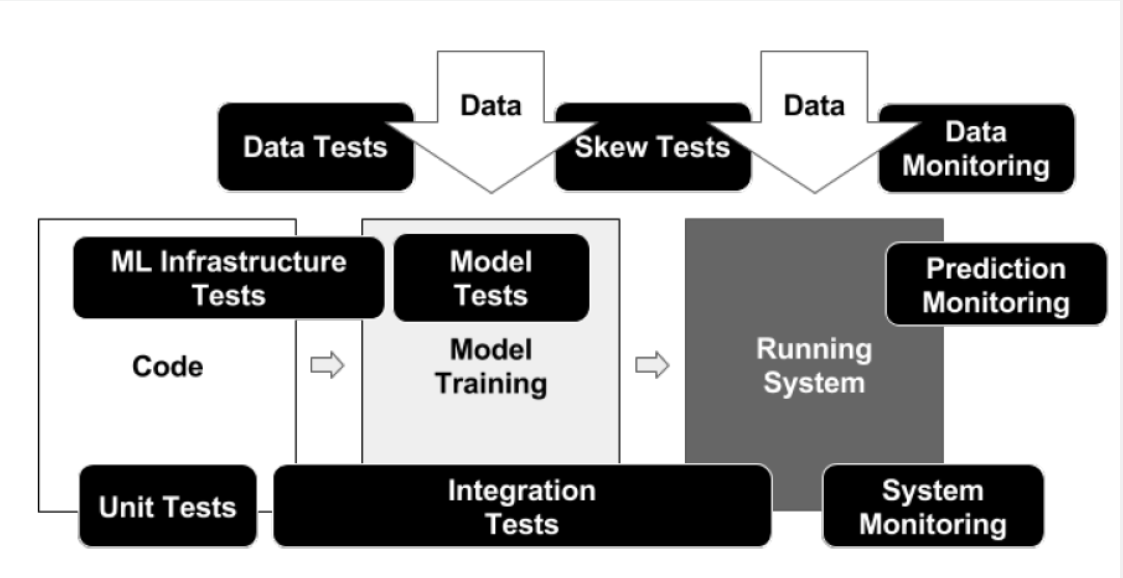
* Sử dụng các kỹ thuật chính quy trong quá trình đào tạo để giảm thiểu vấn đề quên và giữ lại thông tin quan trọng.

#### Secure And Compliant Database:

* Đảm bảo rằng các chiến lược quản lý dữ liệu tuân thủ các quy định an ninh và quy định liên quan đến quyền riêng tư và bảo mật dữ liệu.
* Chiến lược quản lý dữ liệu là một phần quan trọng của quá trình học liên tục và đảm bảo rằng mô hình có khả năng học và duy trì hiệu suất qua thời gian.

## Test Production:

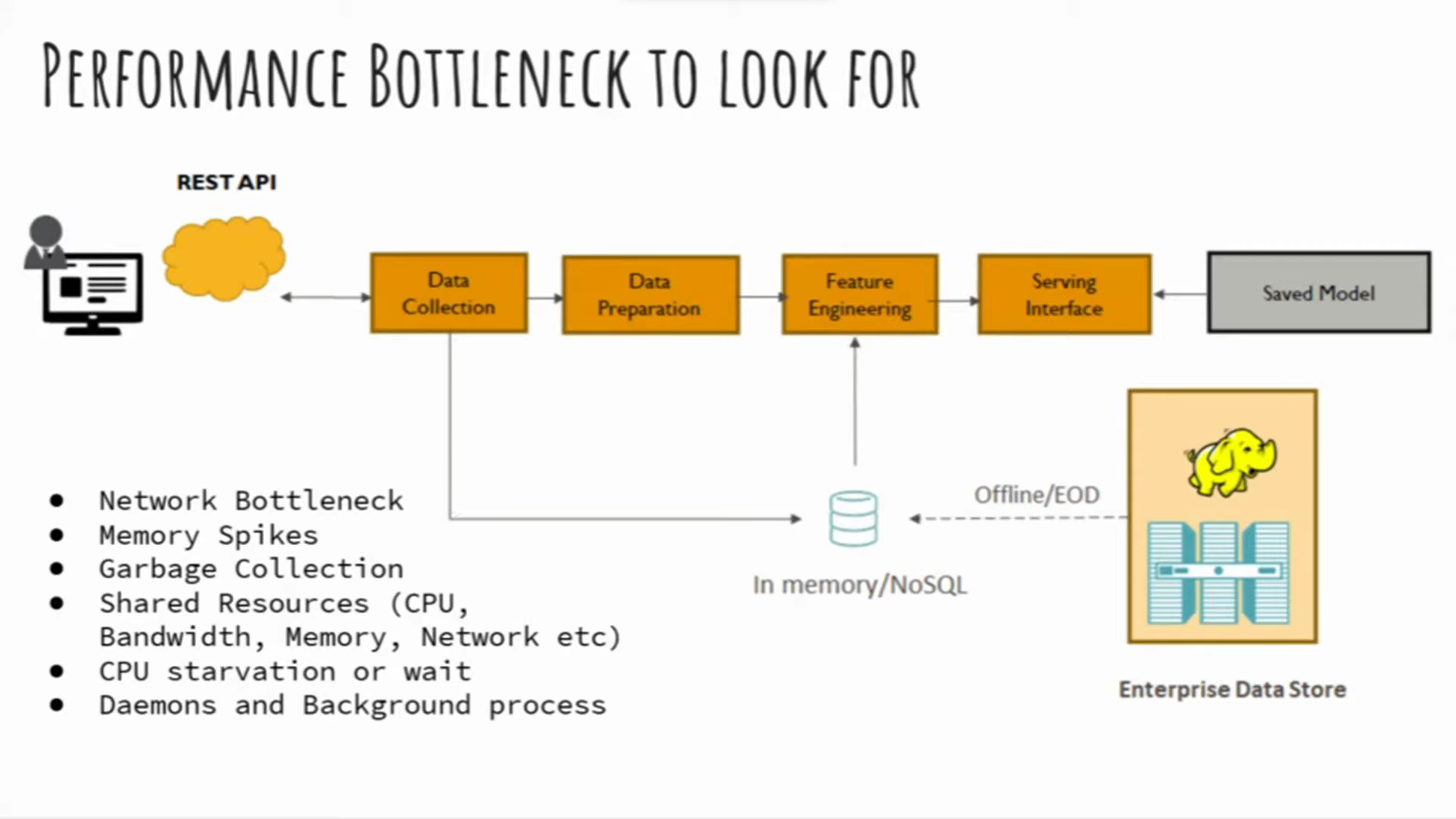
* "Test production" trong machine learning thường ám chỉ quá trình kiểm thử và triển khai mô hình machine learning trong môi trường sản xuất (production environment). Dưới đây là một số khái niệm quan trọng liên quan đến "test production" trong machine learning.



### Model Testing (Kiểm thử mô hình):

* Test set: Đây là một phần của dữ liệu mà mô hình chưa từng được huấn luyện. Tập kiểm thử được sử dụng để đánh giá hiệu suất của mô hình trên các mẫu dữ liệu mới.
* Đánh giá hiệu suất: Để đánh giá hiệu suất của mô hình, các metric như độ chính xác, độ phủ, F1-score, AUC-ROC, và Mean Squared Error (MSE) thường được sử dụng tùy thuộc vào loại vấn đề.
* Cross-Validation: Để đảm bảo tính khách quan của việc đánh giá, phương pháp chéo kiểm tra chia dữ liệu thành các tập con khác nhau và thực hiện kiểm tra trên các tập này. Phổ biến nhất là chia dữ liệu thành k phần bằng nhau (k-fold cross-validation).
* Kiểm thử độ tin cậy: Kiểm tra khả năng của mô hình đối với nhiễu, dữ liệu ngoại lệ hoặc biến đổi về mặt không gian và thời gian. Điều này giúp đảm bảo mô hình hoạt động ổn định trong các điều kiện khác nhau.
* Generalization Testing: Đánh giá khả năng của mô hình áp dụng được cho dữ liệu mới mà nó chưa từng thấy trước đó. Mục tiêu là đảm bảo mô hình không chỉ học thuộc lòng dữ liệu huấn luyện mà còn có khả năng tổng quát hóa đối với dữ liệu mới.
* Performance Testing: Đôi khi cần kiểm tra hiệu suất của mô hình trong môi trường thực tế, đặc biệt là khi triển khai mô hình trên các hệ thống thời gian thực hoặc có dung lượng tính toán hạn chế.
* Safety Testing: Trong các ứng dụng như xe tự lái hay y tế, quan trọng để kiểm tra tính an toàn của mô hình. Điều này bao gồm việc xác định các tình huống biên và kiểm tra cách mô hình ứng xử trong các tình huống đó.
* Isolation Testing: Kiểm tra hiệu suất của mô hình khi được triển khai độc lập, mà không phụ thuộc vào các thành phần bên ngoài.
* Scalability Testing: Kiểm tra khả năng của mô hình khi tăng cường quy mô, đặc biệt là trong môi trường có nhiều người dùng hoặc dữ liệu lớn.
* Quá trình kiểm thử mô hình đòi hỏi sự cân nhắc cẩn thận và chuẩn bị kỹ lưỡng để đảm bảo rằng mô hình không chỉ đạt được hiệu suất tốt trên dữ liệu huấn luyện mà còn có khả năng tổng quát hóa và đáp ứng được các yêu cầu thực tế.

### Kiểm thử hiệu suất (Performance Testing):



Kiểm thử hiệu suất (Performance Testing) trong ngữ cảnh của mô hình máy học và trí tuệ nhân tạo thường tập trung vào đánh giá và đo lường hiệu suất của mô hình trong nhiều khía cạnh khác nhau. Đây là một số khía cạnh quan trọng khi thực hiện kiểm thử hiệu suất:

#### Thời gian dự đoán (Inference Time):

Đo lường thời gian mà mô hình mất để thực hiện dự đoán trên một mẫu dữ liệu mới. Thời gian dự đoán quan trọng trong các ứng dụng thời gian thực như hệ thống nhận diện hình ảnh trên camera an ninh hoặc điều khiển robot.

#### Memory:

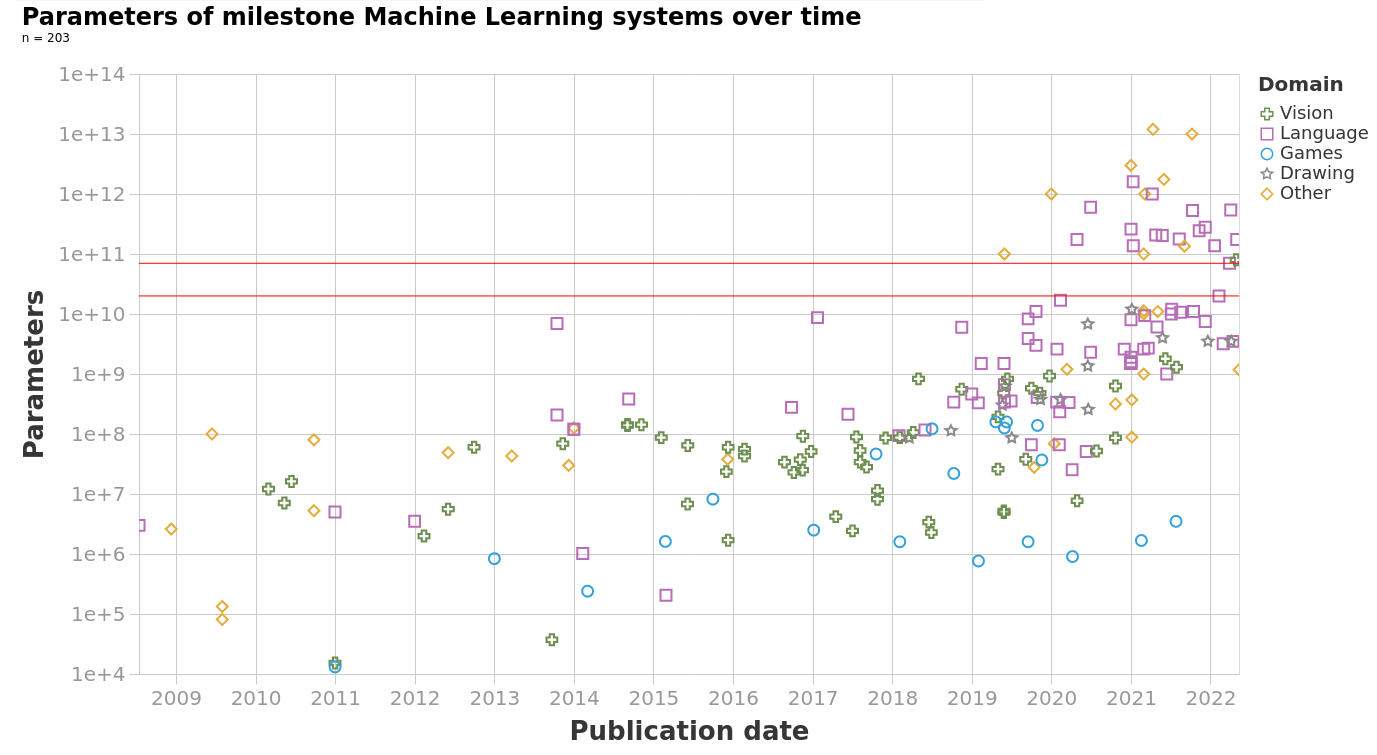
Kiểm tra lượng bộ nhớ mà mô hình sử dụng khi hoạt động. Việc quản lý bộ nhớ quan trọng để đảm bảo mô hình có thể chạy trên các thiết bị có tài nguyên hạn chế.

#### Hiệu năng tính toán:

Đánh giá khả năng tính toán của mô hình, bao gồm việc đo lường số lượng phép toán cần thiết để thực hiện dự đoán. Hiệu năng tính toán đặc biệt quan trọng trong các mô hình phức tạp như các mô hình học sâu.

#### Kích thước của Model:

Kiểm tra kích thước của mô hình, bao gồm số lượng tham số và dung lượng bộ nhớ mà mô hình chiếm giữ. Kích thước mô hình quan trọng đối với việc triển khai trên các thiết bị di động hoặc trong môi trường có dung lượng lưu trữ hạn chế.



#### Kích thước trên dữ liệu lớn:

Đánh giá hiệu suất của mô hình khi đối mặt với dữ liệu lớn. Điều này bao gồm cả thời gian huấn luyện và đánh giá trên các tập dữ liệu có kích thước lớn.

Kiểm tra khả năng của mô hình để mở rộng khi cần xử lý một lượng lớn dữ liệu hoặc đồng thời phục vụ nhiều người dùng.

#### Scalability (khả năng mở rộng):

Kiểm tra khả năng của mô hình để mở rộng khi cần xử lý một lượng lớn dữ liệu hoặc đồng thời phục vụ nhiều người dùng.

#### Safety:

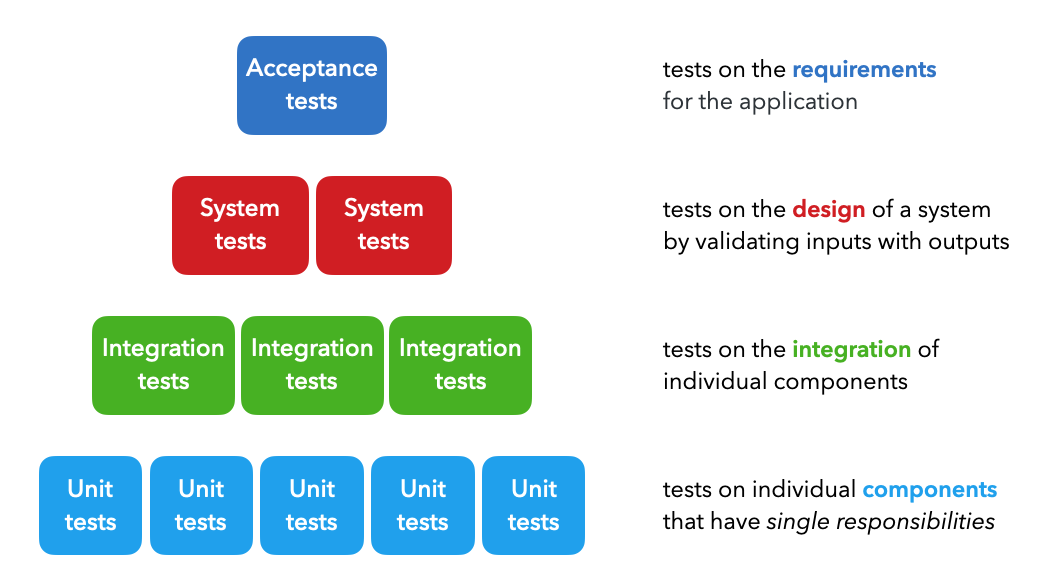
Trong ngữ cảnh của mô hình tự động, đặc biệt là trong các ứng dụng như xe tự lái, việc kiểm tra tính an toàn của mô hình là quan trọng. Điều này bao gồm việc đảm bảo rằng mô hình không chỉ hoạt động hiệu quả mà còn đảm bảo an toàn trong các tình huống biên và khó lường.

#### Kiểm tra độ tin cậy (Reliability Testing):

Đánh giá khả năng của mô hình hoạt động ổn định và đáng tin cậy trong các điều kiện khác nhau, đặc biệt là khi môi trường hoạt động thay đổi.

* *Kiểm thử hiệu suất đóng vai trò quan trọng trong việc đảm bảo rằng mô hình không chỉ đáp ứng được các yêu cầu chức năng mà còn đảm bảo rằng nó có thể được triển khai một cách hiệu quả trong môi trường thực tế.*

### System Testing (kiểm thử hệ thống):



Kiểm thử tích hợp: Đảm bảo rằng các thành phần của hệ thống, bao gồm cả mô hình máy học, được tích hợp và hoạt động một cách đồng nhất. Điều này bao gồm cả kiểm tra tương tác giữa mô hình và các thành phần khác như cơ sở dữ liệu, hệ thống giao diện người dùng, và các dịch vụ khác.

Kiểm thử đồng thời: Đối với hệ thống có nhiều người dùng hoặc có khả năng xử lý đồng thời, kiểm tra khả năng của hệ thống để xử lý nhiều yêu cầu cùng một lúc mà vẫn duy trì hiệu suất và độ chính xác của mô hình.

Kiểm thử độ ổn định: Đảm bảo rằng hệ thống là ổn định và không gặp phải các vấn đề liên quan đến bộ nhớ, rò rỉ tài nguyên, hay các lỗi hệ thống khác khi hoạt động trong môi trường sản xuất.

Kiểm thử độ an toàn: Kiểm tra tính an toàn của hệ thống, đặc biệt là khi liên quan đến xử lý dữ liệu nhạy cảm và đảm bảo rằng mô hình không bị tấn công hoặc lợi dụng.

Kiểm thử phục hồi: Kiểm tra khả năng của hệ thống để phục hồi từ các tình huống sự cố như mất kết nối, sự cố phần cứng, hay sự cố phần mềm.

KIểm thử môi trường: Đảm bảo rằng hệ thống có thể chuyển đổi linh hoạt giữa các môi trường khác nhau, từ môi trường phát triển đến môi trường kiểm thử và cuối cùng là môi trường sản xuất mà không gặp vấn đề không mong muốn.

Kiểm thử hiệu suất: Đánh giá và đo lường hiệu suất của toàn bộ hệ thống, bao gồm cả mô hình máy học và các thành phần khác, để đảm bảo rằng hệ thống có thể đáp ứng được yêu cầu về hiệu suất trong điều kiện thực tế.

Kiểm thử tính toàn vẹn dữ liệu (Data integrity Testing): Kiểm tra tính toàn vẹn của dữ liệu trong quá trình chuyển đổi giữa các thành phần của hệ thống và đảm bảo rằng không có mất mát dữ liệu hay biến đổi không mong muốn.

### Kiểm thử tích hợp (Integration Testing):

Tích hợp mô hình và dữ liệu: Kiểm tra tính đồng bộ giữa mô hình máy học và dữ liệu. Đảm bảo rằng mô hình có thể hiểu và sử dụng đúng đắn dữ liệu được cung cấp từ các nguồn khác nhau và trong các định dạng khác nhau.

Kiểm thử giao tiếp API: Nếu mô hình máy học được tích hợp thông qua API (Interface Lập Trình Ứng Dụng), kiểm tra tính đồng bộ và đúng đắn của các yêu cầu và phản hồi giữa mô hình và các thành phần khác của hệ thống.

Tích hợp cơ sở dữ liệu: Đảm bảo rằng mô hình có thể tương tác với cơ sở dữ liệu một cách đúng đắn. Kiểm tra tính toàn vẹn dữ liệu và đồng bộ hóa dữ liệu giữa cơ sở dữ liệu và mô hình.

Tích hợp hệ thống giao tiếp giao diện người dùng: Nếu mô hình máy học có liên quan đến giao diện người dùng, kiểm tra tính hợp nhất của nó với giao diện người dùng, đảm bảo rằng dữ liệu được truyền đúng và kết quả được hiển thị đúng đắn.

Kiểm thử tích hợp với dịch vụ bên ngoài: Đối với hệ thống sử dụng các dịch vụ bên ngoài, như các dịch vụ máy học hoặc dịch vụ xử lý ngôn ngữ tự nhiên, kiểm tra tích hợp hiệu quả giữa mô hình và các dịch vụ này.

Tích hợp với phần cứng: Nếu mô hình máy học chạy trên phần cứng đặc biệt hoặc tích hợp với các thiết bị nhúng, kiểm tra tích hợp và đúng đắn của mô hình với phần cứng.

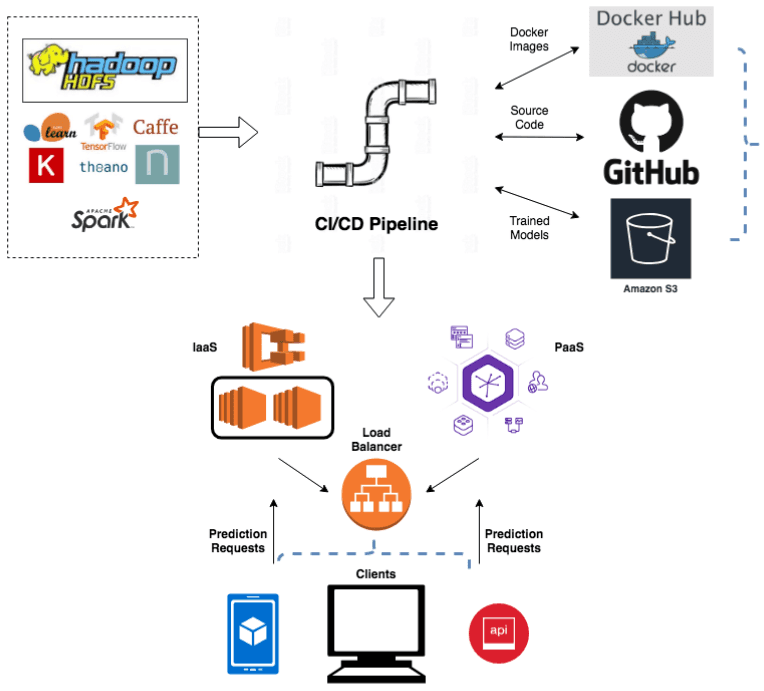
Tích hợp với hệ thống đào tạo (Training System): Đảm bảo tính liên kết giữa mô hình đã được đào tạo và hệ thống đào tạo (training system). Điều này bao gồm việc đảm bảo rằng mô hình sử dụng các trọng số và tham số đúng đắn từ quá trình đào tạo.

Kiểm thử tích hợp các phiên bản mô hình khác nhau: Trong trường hợp cập nhật hoặc triển khai nhiều phiên bản mô hình, kiểm tra tích hợp để đảm bảo rằng các phiên bản này có thể hoạt động cùng nhau mà không gây ra xung đột.

Kiểm thử tích hợp với các hệ thống mô hình phức tạp: Trong các hệ thống sử dụng nhiều mô hình hoặc mô hình phức tạp, kiểm tra tích hợp để đảm bảo rằng các mô hình tương tác một cách đúng đắn và hiệu quả.

* Kiểm thử tích hợp trong machine learning giúp đảm bảo rằng mô hình hoạt động nhất quán và hiệu quả khi tích hợp vào hệ thống tổng thể, và nó đóng vai trò quan trọng trong quá trình chuyển giao mô hình từ môi trường phát triển đến môi trường sản xuất.

### Model Deployment (Triển khai mô hình):



Trong machine learning, "model deployment" (triển khai mô hình) là quá trình chuyển giao một mô hình máy học từ môi trường phát triển (nơi mô hình được huấn luyện và đánh giá) sang môi trường sản xuất (nơi mà mô hình được sử dụng để đưa ra dự đoán trên dữ liệu thực tế). Quá trình triển khai là bước cuối cùng trong quy trình phát triển mô hình và đưa mô hình vào ứng dụng thực tế.

#### Chuẩn bị môi trường:

Chuẩn bị môi trường đòi hỏi sự chú ý đặc biệt đối với các yếu tố cụ thể của mô hình và môi trường triển khai, và quy trình này thường được thực hiện bởi các chuyên gia triển khai và quản trị hệ thống.

**Tài nguyên hệ thống:**

* **Phần cứng**: Xác định và cấu hình tài nguyên phần cứng cần thiết cho việc triển khai mô hình. Điều này bao gồm CPU, GPU (nếu mô hình sử dụng GPU để gia tăng tốc độ tính toán), bộ nhớ, và ổ đĩa lưu trữ.
* **Mạng**: Đảm bảo rằng môi trường triển khai có kết nối mạng ổn định và đủ băng thông để xử lý dữ liệu đầu vào và đầu ra của mô hình.

**Phần mềm hệ thống**

* **Hệ điều hành**: Xác định hệ điều hành được sử dụng trong môi trường triển khai và đảm bảo rằng mô hình được hỗ trợ trên nó. Thông thường, Linux là một lựa chọn phổ biến cho môi trường triển khai.
* **Thư viện và phần mềm hỗ trợ**: Cài đặt và cấu hình các thư viện và phần mềm cần thiết để chạy mô hình, bao gồm cả các thư viện machine learning như TensorFlow, PyTorch, scikit-learn, và các phần mềm khác như Docker để đóng gói mô hình.

**Quản lý tài khoản và Quyền truy cập**: Xác định các tài khoản và quyền truy cập cần thiết để chạy mô hình và quản lý tài nguyên hệ thống. Điều này đảm bảo an toàn và bảo mật trong quá trình triển khai

**Bảo mậ**t: Xác định các biện pháp bảo mật cần thiết để bảo vệ mô hình và dữ liệu. Điều này bao gồm cả việc cài đặt các tường lửa, kiểm soát truy cập, và mã hóa dữ liệu.

**Quản lý phiên bản**: Xác định cách quản lý phiên bản mô hình và code. Việc này có thể bao gồm việc sử dụng các hệ thống quản lý phiên bản như Git và các tiêu chuẩn để đảm bảo quá trình triển khai có thể lặp lại và duy trì được.

**Kiểm thử và xác nhận**: Thực hiện kiểm thử để đảm bảo rằng mô hình và hệ thống hoạt động đúng đắn trong môi trường triển khai dưới các điều kiện thực tế.

#### Chuyển đổi mô hình:

Một số trường hợp đòi hỏi chuyển đổi mô hình từ định dạng sử dụng trong quá trình huấn luyện sang định dạng có thể triển khai, ví dụ như chuyển đổi mô hình từ định dạng TensorFlow sang TensorFlow Lite để sử dụng trên thiết bị di động.

#### Tích hợp với hệ thống:

Mô hình cần được tích hợp vào hệ thống hoạt động, bao gồm cả quá trình tương tác với các thành phần khác của hệ thống.

#### Quản lý dữ liệu đầu vào:

Đảm bảo rằng dữ liệu đầu vào cho mô hình là đúng đắn và được xử lý đúng cách. Điều này có thể bao gồm việc kiểm tra định dạng dữ liệu, xử lý thiếu sót, và chuẩn hóa dữ liệu.

#### Quản lý tài nguyên:

Đảm bảo rằng mô hình có đủ tài nguyên để chạy một cách hiệu quả trong môi trường triển khai, bao gồm cả bộ nhớ, CPU, và GPU nếu cần thiết.

#### Kiểm thử và đánh giá:

Thực hiện kiểm thử kỹ lưỡng để đảm bảo rằng mô hình hoạt động đúng đắn trong môi trường triển khai. Điều này bao gồm cả việc đánh giá hiệu suất và xác định liệu mô hình có đáp ứng đúng yêu cầu hay không.

#### Quản lý mô hình:

Xây dựng cơ sở hạ tầng để quản lý và theo dõi hiệu suất của mô hình trong thời gian thực. Điều này có thể bao gồm việc xây dựng các hệ thống giám sát và ghi nhật ký.

#### Tối ưu hóa và cập nhật:

Theo dõi hiệu suất của mô hình sau khi triển khai và tối ưu hóa nó nếu cần thiết. Ngoài ra, nếu có dữ liệu mới và mô hình cần cập nhật, quá trình cập nhật mô hình cũng cần được xem xét.

### Giám sản và quản lý mô hình (Model Monitoring and Management):

#### Giám sát hiệu suất (performance monitoring):

Giám sát hiệu suất mô hình trong thời gian thực giúp đảm bảo rằng nó vẫn đáp ứng đúng đắn trước các dữ liệu mới và không gặp phải các vấn đề về hiệu suất.

Các chỉ số như độ chính xác, độ phủ, và thời gian đáp trả có thể được theo dõi để xác định liệu mô hình đang thực sự hoạt động hiệu quả hay không.

#### Giám sát dữ liệu đầu vào:

Điều này liên quan đến việc theo dõi dữ liệu đầu vào mà mô hình nhận được. Có thể xảy ra thay đổi trong phân phối dữ liệu hoặc xuất hiện các giá trị ngoại lệ, và giám sát này giúp phát hiện sớm những thay đổi này.

#### Phát hiện sớm vấn đề:

Hệ thống giám sát mô hình cần có khả năng phát hiện sớm mọi vấn đề tiềm ẩn. Điều này bao gồm cả việc cảnh báo khi có giảm hiệu suất đột ngột hoặc khi có dấu hiệu của sự thay đổi trong dữ liệu.

#### Quản lý phiên bản mô hình:

Để theo dõi và quản lý các phiên bản của mô hình, một hệ thống phiên bản hóa cần được thiết lập. Điều này giúp theo dõi những thay đổi trong mô hình và dễ dàng quay trở lại phiên bản trước nếu cần.

#### Tự động hóa quy trình duy trì:

Việc tự động hóa quy trình duy trì giúp giảm thiểu sai sót và đảm bảo rằng mọi cập nhật và điều chỉnh được thực hiện một cách liền mạch và hiệu quả.

Điều này bao gồm cả việc triển khai các mô hình mới và cập nhật các phần mềm liên quan.

#### Bảo mật và quản lý quyền:

Bảo mật là yếu tố quan trọng trong giám sát và quản lý mô hình. Điều này bao gồm việc kiểm soát quyền truy cập vào mô hình và dữ liệu, bảo vệ khỏi các mối đe dọa an ninh.

#### Giám sát liên tục:

Thay vì chỉ giám sát trong giai đoạn triển khai, quá trình giám sát mô hình cần được thực hiện liên tục, với các cập nhật thường xuyên để theo dõi sự thay đổi trong môi trường và dữ liệu.

#### Quản lý dữ liệu và chuẩn hóa:

Việc quản lý dữ liệu một cách chặt chẽ và chuẩn hóa giúp đảm bảo rằng mô hình có thể đáp ứng mọi biến động trong dữ liệu và giữ được độ chính xác cao.

---------------------------------------------HẾT----------------------------------------------