Họ và tên: **Lê Khắc Thanh Tùng**

MSSV: **52100943**

# **TÌM HIỂU VÀ SO SÁNH CÁC PHƯƠNG PHÁP OPTIMIZER TRONG HUẤN LUYỆN MÔ HÌNH HỌC MÁY**

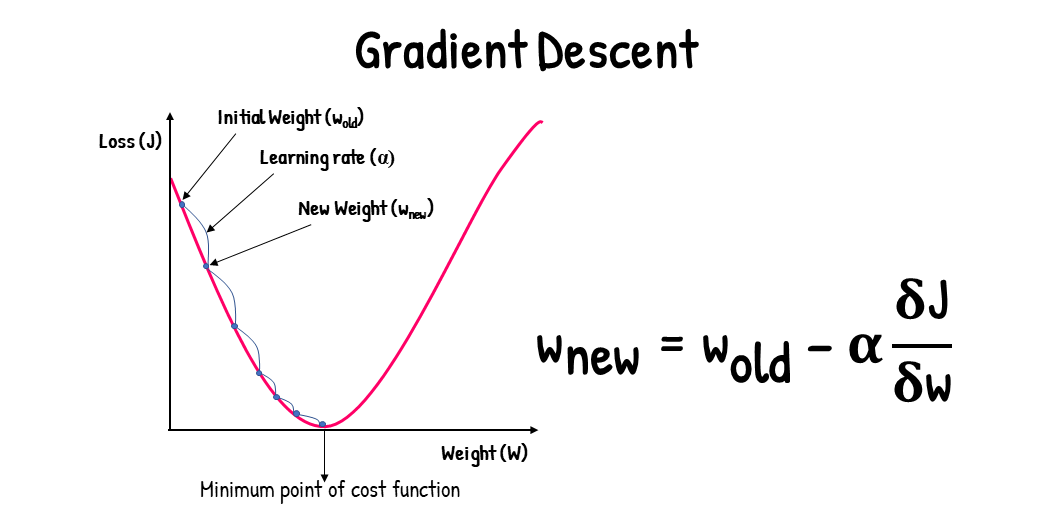
## **Optimizer là gì**

Thuật toán tối ưu (optimizers) đơn giản là ta tìm tham số tối ưu cho mô hình để loss function đạt được giá trị nhỏ nhất và cải thiện độ chính xác khi dự đoán của mô hình.

Cơ chế hoạt động của optimizer thường dựa trên các phương pháp tối ưu hóa đạo hàm để điều chỉnh các tham số của mô hình. Quá trình này thường được thực hiện thông qua việc cập nhật các tham số dựa trên đạo hàm của loss function và một số thông số điều chỉnh như learning rate, epoch,....

## **Các thuật toán optimizer**

### ***Gradient Descent (GD)***



Thuật toán GD bắt đầu bằng việc chọn một điểm khởi tạo ngẫu nhiên trong không gian tham số. Sau đó, nó tính gradient của loss function tại điểm đó.

Tiếp theo, GD điều chỉnh các tham số theo hướng ngược với gradient, với một learning rate nhất định. Learning rate quyết định độ lớn của bước đi và có thể ảnh hưởng đến tốc độ hội tụ.

Quá trình cập nhật gradient và điều chỉnh tham số được lặp lại cho đến khi đạt được điều kiện dừng như đạt được số lượng vòng lặp tối đa hoặc loss function không còn giảm nữa.

**Ưu điểm**:

* Dễ triển khai và hiệu quả tính toán.

**Nhược điểm**:

* GD có thể hội tụ chậm đối với các mô hình có không gian tham số lớn.
* Phụ thuộc vào nghiệm khởi tạo ban đầu và learning rate.
* Learning rate quá lớn sẽ khiến cho thuật toán không hội tụ được vì bước nhảy quá lớn hoặc learning rate nhỏ ảnh hưởng đến tốc độ training

### ***Stochastic Gradient Descent (SGD)***

Công thức toán học:



SGD là 1 biến thể của Gradient Descent .Thay vì sau mỗi epoch chúng ta sẽ cập nhật weight 1 lần thì trong mỗi epoch có N điểm dữ liệu chúng ta sẽ cập nhật trọng số N lần.

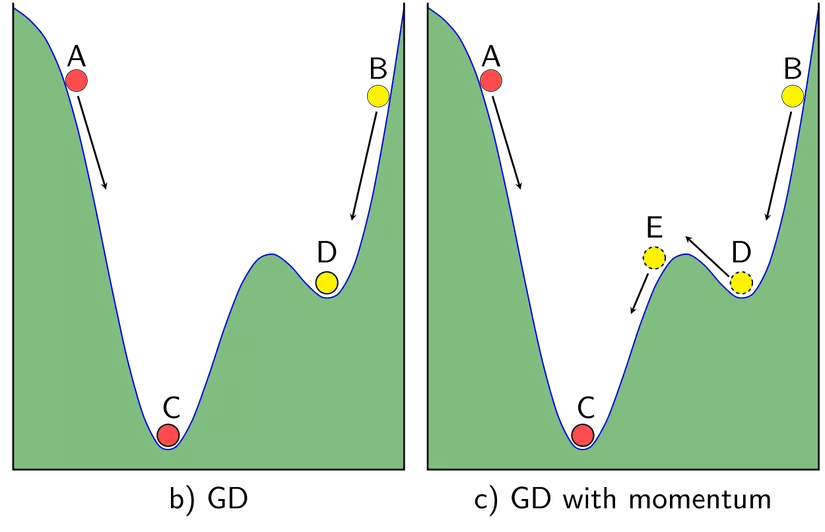
**Ưu điểm**:

* Hiệu quả cho dữ liệu lớn.

**Nhược điểm**:

* Vẫn chưa giải quyết được 2 nhược điểm lớn của gradient descent (learning rate, điểm dữ liệu ban đầu)

### ***Momentum***



Như hình b phía trên, nếu ta thả 2 viên bi tại 2 điểm khác nhau A và B thì viên bị A sẽ trượt xuống điểm C còn viên bi B sẽ trượt xuống điểm D, nhưng ta lại không mong muốn viên bi B sẽ dừng ở điểm D (local minimum) mà sẽ tiếp tục lăn tới điểm C (global minimum). Để thực hiện được điều đó ta phải cấp cho viên bi B 1 vận tốc ban đầu đủ lớn để nó có thể từ điểm D vượt qua điểm E tới điểm C.

Công thức toán học:



**Ưu điểm**:

* Giúp vượt qua các điểm local minimum và hướng tới các điểm global minimum nhanh hơn.

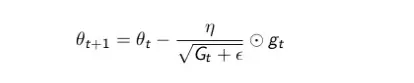
**Nhược điểm**:

* Tới gần đích vẫn mất khá nhiều thời gian giao động qua lại trước khi dừng hẳn

### ***Adagrad***

Các thuật toán trước learning rate đều là hằng số tuy nhiên trong thuật toán Adagrad coi learning rate là 1 tham số nghĩa là Adagrad sẽ cho cập nhật learning rate sau mỗi thời điểm t.

Công thức toán học:



Trong đó :

n : hằng số

gt : gradient tại thời điểm t

ϵ : hệ số tránh lỗi ( chia cho mẫu bằng 0)

G : là ma trận chéo mà mỗi phần tử trên đường chéo (i,i) là bình phương của đạo hàm vectơ tham số tại thời điểm t.

**Ưu điểm**:

* Không cần điều chỉnh learning rate bằng tay

**Nhược điểm**:

* Tổng bình phương biến thiên sẽ lớn dần theo thời gian cho đến khi nó làm learning rate cực kì nhỏ dẫn đến training chậm

### ***RMSprop***

RMSprop giải quyết vấn đề learning rate giảm dần của Adagrad bằng cách chia learning rate cho trung bình của bình phương gradient.

Công thức toán học:



**Ưu điểm**:

* Giải quyết được vấn đề tốc độ học giảm dần của Adagrad

**Nhược điểm**:

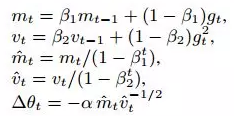
* Nghiệm tìm được chỉ là local minimum chứ không đạt được global minimum như Momentum

### ***Adam (Adaptive Moment Estimation)***

Thuật toán Adam là sự kết hợp của Momentum và RMSprop.

Dưới góc nhìn vật lý 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 vì nó có ma sát nên dễ dừng lại hơn.

Công thức toán học:



**Ưu điểm**:

* Hiệu quả trong việc tối ưu hóa và hội tụ nhanh chóng.
* Phù hợp cho nhiều loại mô hình học máy vì thế đa phần các mô hình học máy hiện nay người ta dùng thuật toán Adam

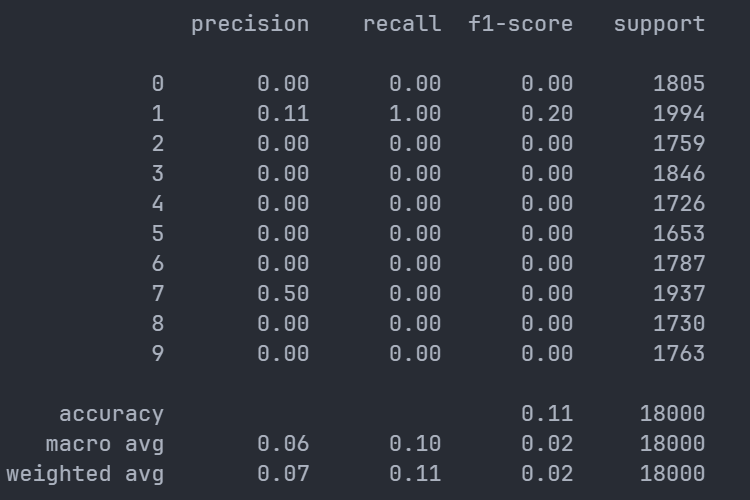
## **So sánh**

Em sẽ xây dựng neural network model với các loại optimizer ở trên để so sánh độ chính xác của các mô hình.

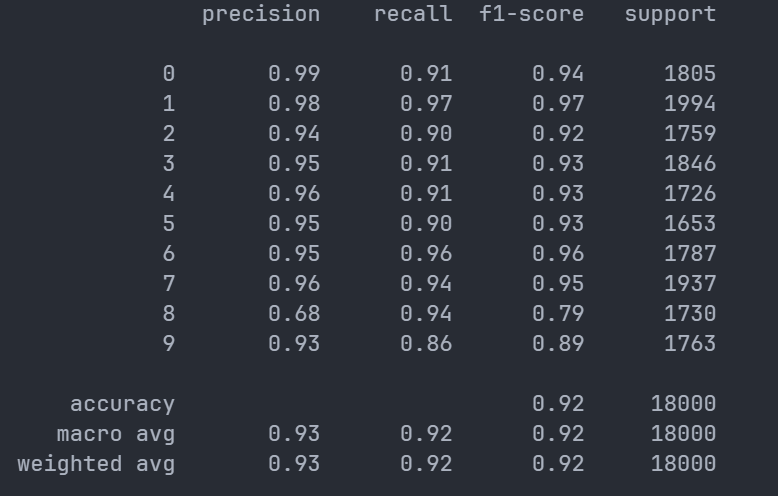
Dataset: <https://www.kaggle.com/datasets/oddrationale/mnist-in-csv>

Optimizer sử dụng:

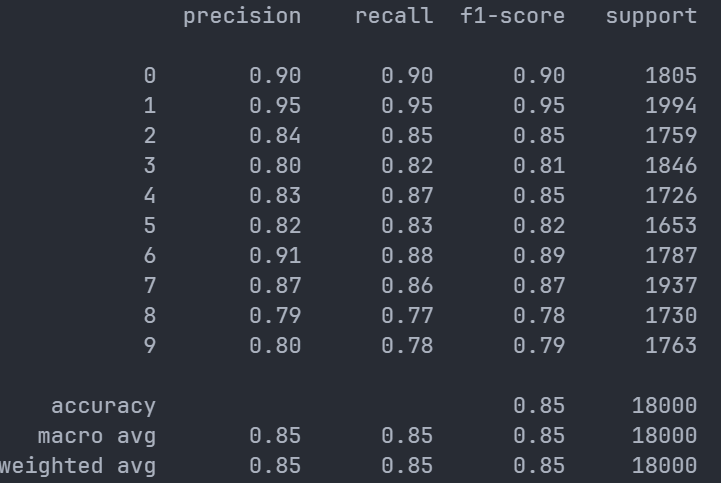
* SGD



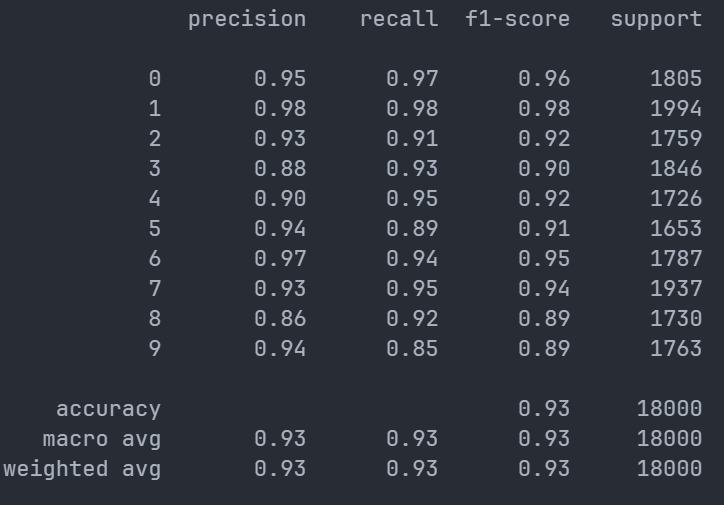
* RMSprop



* Adagrad



* Adam



**Lưu ý:** đây chỉ là so sánh trong một trường hợp (dataset) cụ thể không phải là so sánh tổng quát nên có thể một vài optimizer không phù hợp cho dataset này. Mục đích so sánh trong trường hợp cụ thể chỉ để chỉ rõ ra sự chênh lệch trong mức độ chính xác.

Tổng quan thì optimizer Adam có độ chính xác cao nhất.

# **TÌM HIỂU VỀ CONTINUAL LEARNING VÀ TEST PRODUCTION KHI XÂY DỰNG MỘT GIẢI PHÁP HỌC MÁY ĐỂ GIẢI QUYẾT MỘT BÀI TOÁN NÀO ĐÓ**

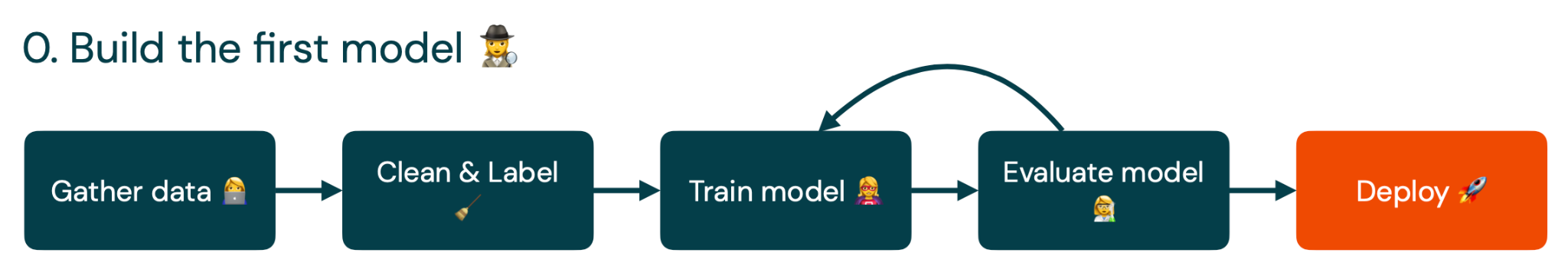
## **Continual Learning**

### ***Tổng quan***

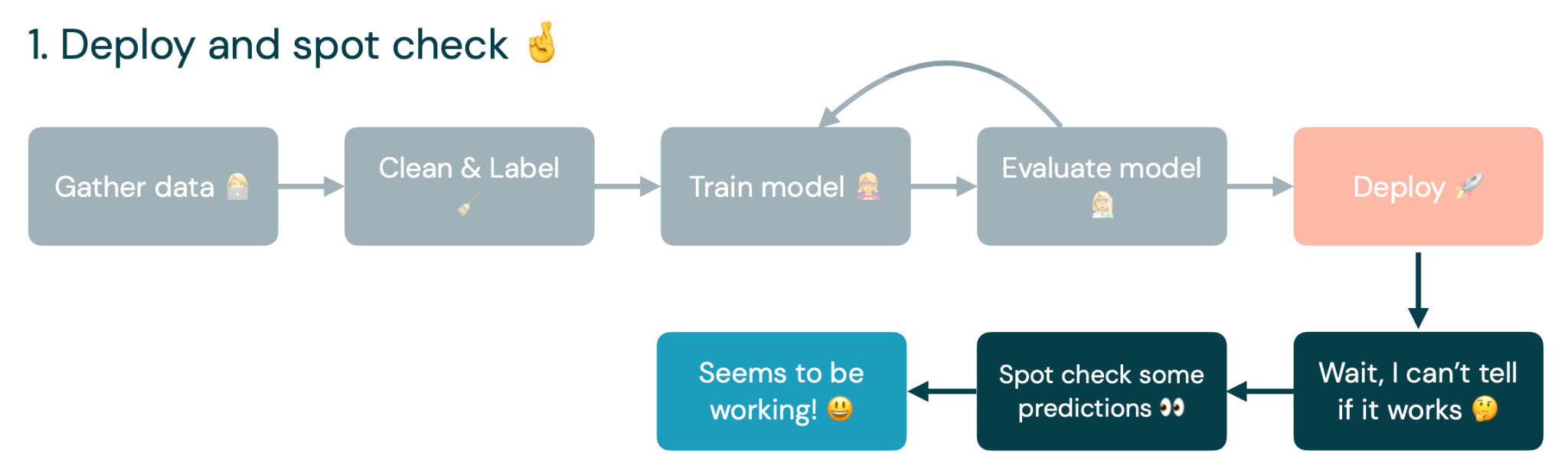
Mô hình học máy nếu muốn sử dụng thực tế thì phải được train liên tục để giảm thiểu sự sai sót khi dự đoán. Vì vậy không thể nào deploy mô hình đã train trước đó lên mà không tiếp tục train cho nó

Ví dụ: một nền tảng xã hội như Facebook, mỗi khi người dùng like một bài post nào đó thì đó sẽ là data đầu vào cho mô hình được train để có thể dự đoán rằng là người dùng có xu hướng thích các bài post thể loại như vậy để những lần sau có thể đề xuất tương tự.

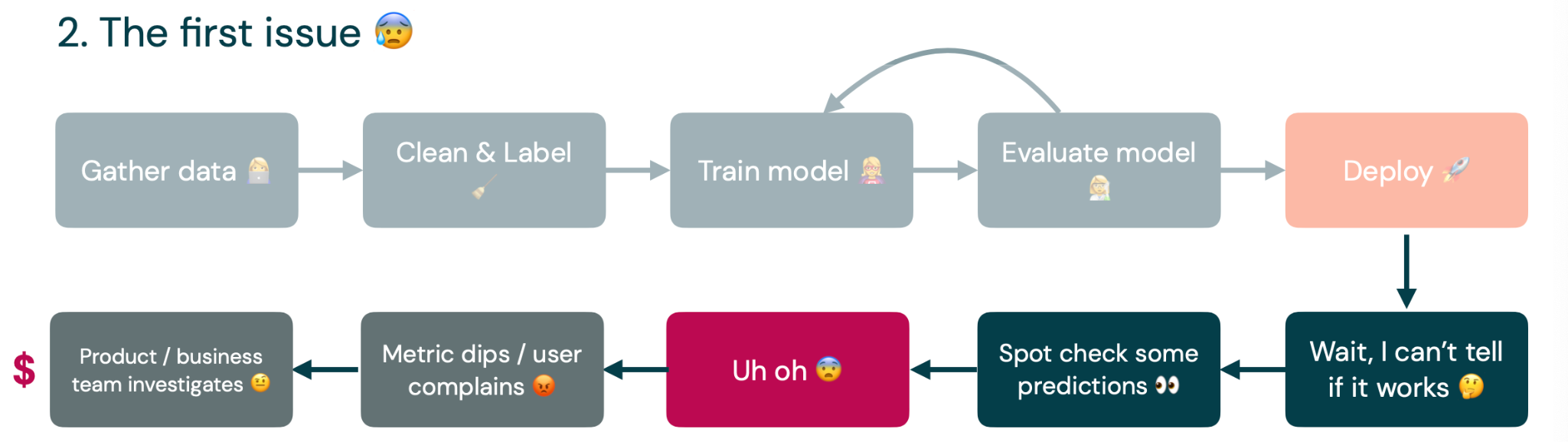
Tóm lại là sẽ luôn có dữ liệu mới và ta sẽ dùng dữ liệu đó để train tiếp cho model.



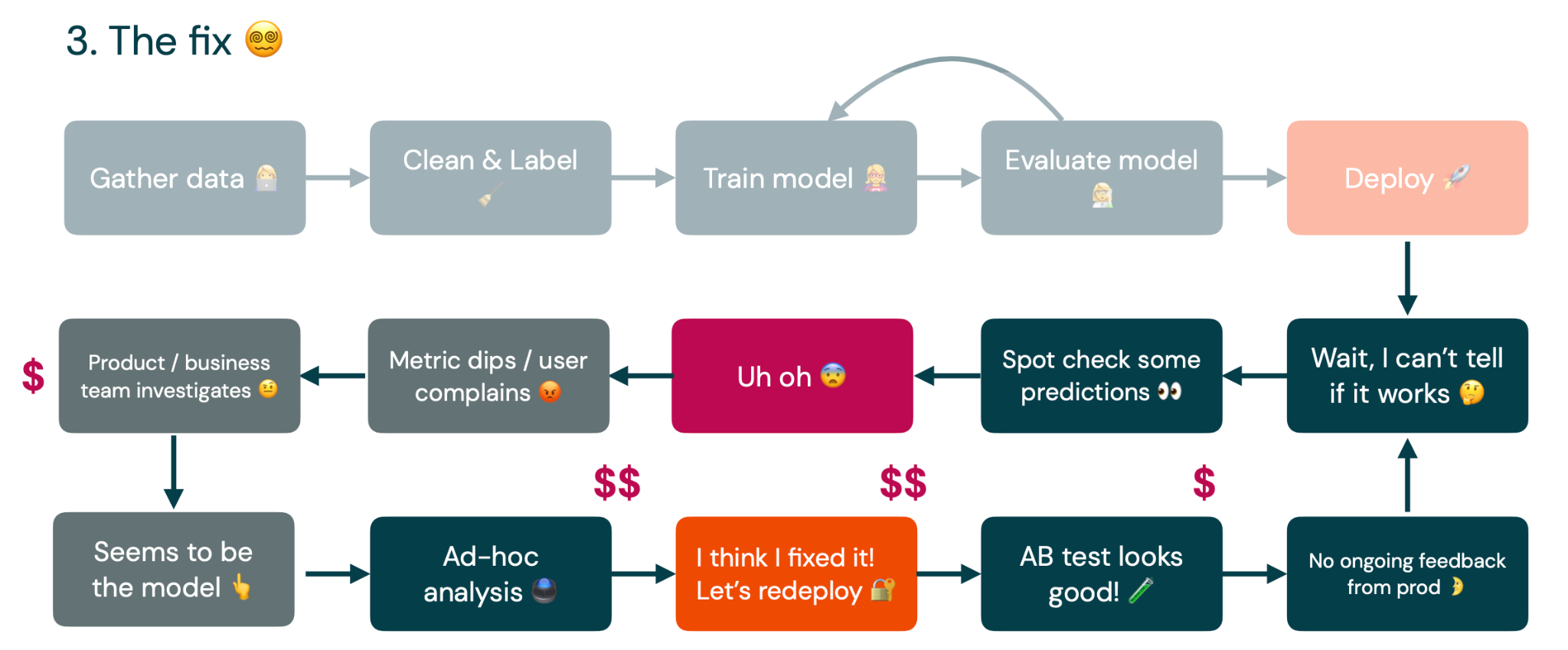
Đây là các bước cơ bản để xây dựng model từ thu thập dataset, pre-processing data, train cho model và đánh giá chất lượng model cho đến deploy lên server nào đó để sử dụng.



Sau khi đã deploy lên server, ta thử kiểm tra mô hình dự đoán ra kết quả như thế nào thì ta thấy model vẫn dự đoán rất tốt.



Tuy nhiên do có một vài trường hợp mà model chưa được train qua bao giờ dẫn đến dự đoán sai.



Giải pháp là train model tiếp rồi deploy lại sau đó kiểm tra mô hình. Quy trình này cứ lặp đi lặp lại để model trở nên tốt hơn.

### ***Ưu điểm***

* Mô hình sẽ mang tính tổng quát hóa hơn không có bias cho trường hợp cụ thể nào
* Hữu ích cho các loại dữ liệu luôn thay đổi theo thời gian
* …

### ***Hạn chế***

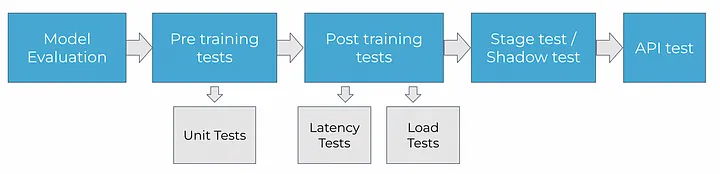
Một trong những hạn chế của continual learning khá giống con người đó là catastrophic forgetting. Khi một mô hình học máy được huấn luyện trên một tập dữ liệu và sau đó được huấn luyện trên tập dữ liệu mới, nó có thể quên đi những kiến thức đã học trước đó. Điều này làm giảm hiệu suất của mô hình.

### ***Giải pháp***

Để giải quyết các hạn chế này, các phương pháp đã được đề xuất như Elastic Weight Consolidation, Synaptic Intelligence và nhóm phương pháp dựa trên Generative Replay. Những phương pháp này tập trung vào việc duy trì kiến thức đã học trước đó trong mô hình và kiểm soát quá trình học tiếp tục để giảm thiểu quên và nhiễu.

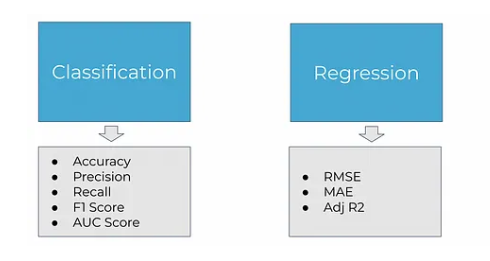
## **Test Production**

Khác với khi kiểm thử trên ứng dụng truyền thống ta kiểm tra xem logic của nghiệp vụ có hoạt động chính xác không thì khi kiểm thử với mô hình học máy ta kiểm tra learned logic.



Các bước kiểm thử mô hình học máy

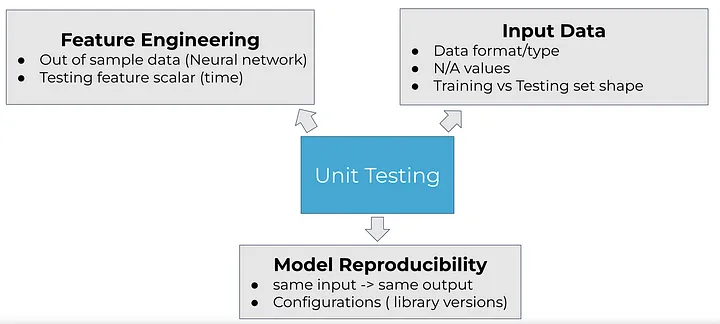
### ***Model Evaluation***



Trước khi train mô hình, ta tách tập dữ liệu ra thành tập dùng để train và tập dùng để test. Sau đó ta dùng mô hình để dự đoán trên tập dữ liệu test và kiểm tra độ chính xác của dự đoán. Ngoài ra ta cũng có thể kiểm tra mô hình có bị overfit hay underfit hay không ở bước kiểm thử này.

### ***Pre-training test***

which can be run without trained parameters—check if our written logic is correct

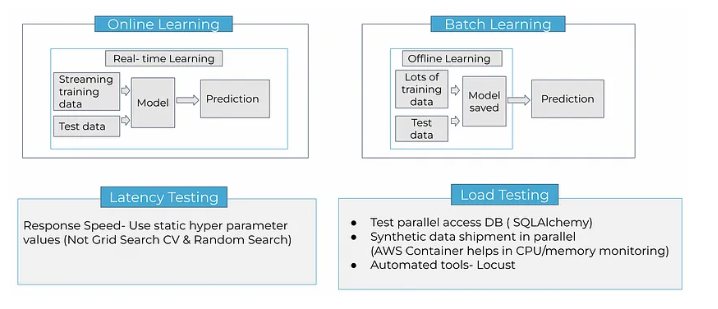


Được chạy trước khi mô hình được train.

Ví dụ:

* Kiểm tra xem đã xử lý các dữ liệu NaN, empty chưa
* Kiểm tra version các package
* …

### ***Post-training test***



Kiểm tra learned logic có như mong đợi, độ chính xác, độ trễ của mô hình, độ chịu tải của mô hình,...

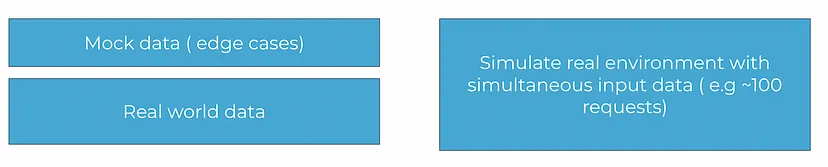
Ví dụ:

* Học sinh học nhiều thì có xu hướng qua môn.
* Nếu có nhiều request đồng thời thì mô hình có xử lý được không
* …

**A/B testing: Model retraining**

Ví dụ: dữ liệu được train cho mô hình là từ năm 2015 tuy nhiên đối với dữ liệu năm 2023 thì các dự đoán nó không còn khớp nữa. Vì vậy ta xây dựng một mô hình mới với dữ liệu năm 2023 và so sánh với mô hình cũ xem hiệu suất mô hình nào tốt hơn thì ta sử dụng mô hình đó.

### ***Stage test/ Shadow test***

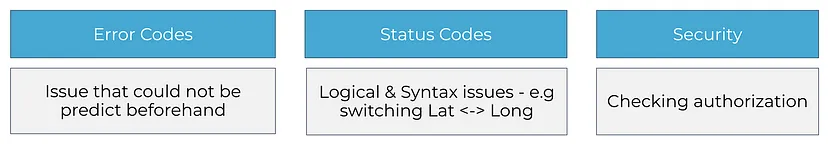


Staging test là kiểm thử mô hình sau khi deploy mô hình với dữ liệu thử nghiệm đa dạng tương tự với các đặc điểm dữ liệu trong thế giới thực.

Shadow Testing:

Shadow testing là một phương pháp kiểm thử trong đó mô hình mới được triển khai và chạy song song với mô hình hiện có. Dữ liệu mới được chạy qua cả hai mô hình và kết quả được so sánh để đánh giá hiệu suất và tính đáp ứng của mô hình mới.

### ***API test***



Đơn giản là khi người dùng request thì có nhận được response không.

Tài liệu tham khảo:

<https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8>

[Thuật toán tối ưu adam (viblo.asia)](https://viblo.asia/p/thuat-toan-toi-uu-adam-aWj53k8Q56m)

<https://fullstackdeeplearning.com/course/2022/lecture-6-continual-learning/>

<https://medium.com/bright-ml/machine-learning-artificial-intelligence-testing-for-production-c561590ff292>

[Machine Learning in Production - Testing (applyingml.com)](https://applyingml.com/resources/testing-ml/)