**TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙢⬩🙠

***Tìm hiểu TensorFlow***

***và nhận dạng số viết tay***

***Giảng viên hướng dẫn:*** TS Nguyễn Đình Hiển

***Nhóm sinh viên thực hiện:***

1. Cao Tấn Huy 43.01.103.020
2. Lâm Phát Tài 44.01.104.188
3. Nguyễn Xuân Tính 44.01.104.196

**Lời mở đầu**

Hiện nay, với hệ thống dữ liệu hình ảnh khổng lồ trên toàn thế giới, việc phân tích, xử lý để khai thác sử dụng thông tin trong ảnh là một nhu cầu thiết yếu. Tuy nhiên, với số lượng dữ liệu khổng lồ, việc phân tích, xử lý thủ công sẽ mất rất nhiều thời gian và nguồn nhân lực. Thực tiễn đã chứng minh, việc ứng dụng trí tuệ nhân tạo nói chung hay Deep Learning nói riêng vào các nhiệm vụ trên đã giúp tiết kiệm được nhiều thời gian và công sức. Từ giữa năm 2011, Google đã giới thiệu dự án Deep Learning sử dụng mạng neuron nhân tạo dùng cho nhận dạng giọng nói và sau đó mở rộng lên các lĩnh vực khác như Gmail, Google dịch, Google ảnh …. Đối với bài toán nhận dạng chữ số viết tay, mạng neuron nhiều lớp sẽ được huấn luyện dựa trên các pixel, đơn vị nhỏ nhất của hình ảnh. Vì vậy mạng neuron nhân tạo là công cụ vô cùng thích hợp cho việc xử lý, phân tích hình ảnh và mang lại kết quả rất khả quan. Nhận dạng chữ số viết tay dùng mạng neuron nhân tạo.

Mục tiêu báo cáo này là trình bày các khái niệm căn bản, xây dựng mô hình nhận diện chữ số viết tay bằng mạng neuron nhân tạo cho tỷ lệ nhận dạng đúng trên 90%. Các phần tiếp theo sẽ chỉ ra đơn vị cấu tạo nhỏ nhất tạo thành mạng neuron nhân tạo đa lớp và cấu trúc, thông số của mô hình mạng neuron đa lớp cũng như kết quả thực nghiệm với các trường hợp khác nhau để khảo sát, làm rõ chức năng của các thông số trong mô hình mạng.

**MỤC LỤC**

[I. Giới thiệu về TensorFlow 1](#_Toc56462012)

[1.1 TensorFlow là gì ? 1](#_Toc56462013)

[1.2 Lịch sử ra đời 1](#_Toc56462014)

[1.2.1 DistBelief 1](#_Toc56462015)

[1.2.2 TensorFlow 2](#_Toc56462016)

[1.3 Những khái niệm cơ bản về TensorFlow 2](#_Toc56462017)

[1.3.1 Tensor 2](#_Toc56462018)

[1.3.2 Rank 3](#_Toc56462019)

[1.3.3 Shape 3](#_Toc56462020)

[1.3.4 Type 3](#_Toc56462021)

[1.4 . Cách TensorFlow hoạt động 4](#_Toc56462022)

[1.5 Lợi ích 4](#_Toc56462023)

[II. Nhận dạng chữ số viết tay dùng mạng neuron nhân tạo 6](#_Toc56462024)

[2.1 Neuron nhân tạo 6](#_Toc56462025)

[2.1.1 Trọng số (Weight) 6](#_Toc56462026)

[2.1.2 Bias 7](#_Toc56462027)

[2.1.3 Neuron sigmoid và hàm sigmoid 7](#_Toc56462028)

[2.2. Mô hình mạng Neuron nhân tạo 8](#_Toc56462029)

[2.2.1 Cấu trúc mạng neuron nhân tạo 8](#_Toc56462030)

[2.2.2 Phương pháp “học” trong mạng MLP 9](#_Toc56462031)

[III. Chương trình nhận dạng chữ số và phép tính cơ bản 10](#_Toc56462032)

[3.1 Mô tả hoạt động các hàm 10](#_Toc56462033)

[3.2. Hướng dẫn sử dụng 13](#_Toc56462034)

# I. Giới thiệu về TensorFlow

## 1.1 TensorFlow là gì ?

TensorFlow là một thư viện phần mềm mã nguồn mở dành cho máy học trong nhiều loại hình tác vụ nhận thức và hiểu ngôn ngữ.

Nó hiện đang được sử dụng cho cả nghiên cứu lẫn sản xuất bởi 50 đội khác nhau trong hàng top 2 sản phẩm thương mại của Google, như nhận dạng giọng nói, Gmail, Google Photos, và tìm kiếm, nhiều trong số đó đã từng sử dụng chương trình tiền nhiệm DistBelief của nó. TensorFlow nguyên thủy được phát triển bởi đội Google Brain cho mục đích nghiên cứu và sản xuất của Google và sau đó được phát hành theo giấy phép mã nguồn mở Apache

## 1.2 Lịch sử ra đời

### 1.2.1 DistBelief

Bắt đầu từ năm 2011, Google Brain xây dựng DistBelief như là hệ thống máy học thế hệ đầu tiên, độc quyền của mình. Hơn 50 đội tại Google và các công ty Alphabet khác đã triển khai các mạng nơ-ron học sâu của DistBeliistBelief trong các sản phẩm thương mại của Google, bao gồm Google Search,Google Voice Search, quảng cáo, Google Photos, Google Maps, Google Street View, Google Translate, và YouTube. Google đã chỉ định các nhà khoa học máy tính, như tiến sĩ Geoffrey Hinton và tiến sĩ Jeff Dean, để đơn giản hóa và cải tiến mã nguồn codebase của DistBelief để trở thành một thư viện lớp ứng dụng nhanh hơn, mạnh mẽ hơn, mà sau này trở thành TensorFlow. Trong năm 2009, đội ngũ được lãnh đạo bởi Hinton đã có thể giảm số lỗi trong các mạng nơ-ron sử dụng DistBelief một lượng đáng kể; đột phá này đã được thực hiện bởi các đột phá khoa học của Hinton trong truyền ngược tổng quát hóa. Đột phá đáng chú ý nhất của Hinton trực tiếp dẫn đến việc giảm lỗi trong phần mềm nhận dạng giọng nói của Google ít nhất là 25%.

### 1.2.2 TensorFlow

TensorFlow là thế hệ thứ hai của hệ thống học máy của Google Brain, với một bản cài đặt tham khảo đã phát hành dưới dạng phần mềm mã nguồn mở vào ngày 9/11/2015. Trong khi bản cài đặt tham khảo chạy trên một thiết bị đơn, TensorFlow có thể chạy trên nhiều CPU và GPU (với nhiều mở rộng CUDA tùy chọn cho việc tính toán đa năng trên các GPU). Nó chạy trên desktop Linux hoặc Mac OS X 64-bit hoặc các hệ thống máy chủ, cũng như trên các nền tảng điện toán di động, bao gồm Android và iOS của Apple. Các tính toán của TensorFlow được thể hiện dưới dạng các biểu đồ dataflow chi tiết. Nhiều nhóm tại Google đã chuyển từ DistBelief sang TensorFlow để phục vụ cho việc nghiên cứu và sản xuất .Thư viện thuật toán này bắt nguồn từ nhu cầu của Google để hướng dẫn (lập trình) các hệ thống máy tính, được gọi là mạng nơron, để tìm hiểu và lý luận tương tự cách thức của con người, vì vậy mà các ứng dụng mới có thể được xuất phát từ đây có thể đảm nhận các vai trò và chức năng vốn trước đây chỉ dành cho những người có khả năng; cái tên TensorFlow bản thân nó xuất phát từ các thao tác mà các mạng nơron như vậy thực hiện trên các mảng dữ liệu đa chiều. Những mảng đa chiều được gọi là các "tensor" nhưng khái niệm này là không giống với khái niệm 'tensor' trong toán học .Mục đích là để huấn luyện các mạng nơron phát hiện và giải mã các mẫu và các mối tương quan.

Vào tháng 6/2016, Jeff Dean của Google cho biết đã có 1.500 repository (kho) trên GitHub đề cập tới TensorFlow, trong đó chỉ có 5 đến từ Google.

## 1.3 Những khái niệm cơ bản về TensorFlow

### 1.3.1 Tensor

Tensor là cấu trúc dữ liệu trong TensorFlow đại diện cho tất cả các loại dữ liệu. Nói cách khác, tất cả các kiểu dữ liệu khi đưa vào trong TensorFlow thì đều được gọi là Tensor. Vậy nên có thể hiểu được Tensorflow là một thư viện mô tả, điều chỉnh dòng chảy của các Tensor.

Tensor có 3 thuộc tính cơ bản là *rank*, *shape* và *type*.

### 1.3.2 Rank

Rank là số bậc của Tensor.

Ví dụ Tensor = [1] thì có rank = 1; Tensor = [[3,4],[5,6]] thì sẽ có rank = 2. Việc phân rank này khá quan trọng vì nó đồng thời cũng giúp phân loại dữ liệu của Tensor.

Khi các rank đặc biệt cụ thể, Tensor có những tên gọi riêng như sau:

Rank = 0 → Scalar

Rank = 1 → Vector

Rank = 2 → Matrix

Rank > 2 → N-Tensor

### 1.3.3 Shape

Shape là chiều của Tensor.

Ví dụ: Tensor = [[[1,1,1],[178,62,74]]] sẽ có Shape = (1,2,3)

Tensor = [[1,1,1],[178,62,74]] sẽ có Shape = (2,3).

### 1.3.4 Type

Type kiểu dữ liệu của các elements trong Tensor. Vì một Tensor chỉ có duy nhất một thuộc tính Type nên từ đó cũng suy ra là chỉ có duy nhất một kiểu Type duy nhất cho toàn bộ các elements có trong Tensor hiện tại.

## 1.4 . Cách TensorFlow hoạt động

TensorFlow cung cấp tất cả những điều này cho lập trình viên theo phương thức của ngôn ngữ Python. Vì Python khá dễ học và làm việc, ngoài ra còn cung cấp nhiều cách tiện lợi để ta hiểu được làm thế nào các high-level abstractions có thể kết hợp cùng nhau. Node và Tensor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow bản thân chúng cũng là các ứng dụng Python.

Các operation toán học thực sự thì không được thi hành bằng Python. Các thư viện biến đổi có sẵn thông qua TensorFlow được viết bằng các binary C++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các high-level abstraction lập trình để nối chúng lại với nhau.

TensorFlow 2.0, được ra mắt vào tháng 10 năm 2019, cải tiến framework theo nhiều cách dựa trên phản hồi của người dùng, để dễ dàng và hiệu quả hơn khi làm việc cùng nó (ví dụ: bằng cách sử dụng các Keras API liên quan đơn giản cho việc train model). Train phân tán dễ chạy hơn nhờ vào API mới và sự hỗ trợ cho TensorFlow Lite cho phép triển khai các mô hình trên khá nhiều nền tảng khác nhau. Tuy nhiên, nếu đã viết code trên các phiên bản trước đó của TensorFlow thì bạn phải viết lại, đôi lúc 1 ít, đôi lúc cũng khá đáng kể, để tận dụng tối đa các tính năng mới của TensorFlow 2.0.

## 1.5 Lợi ích

Lợi ích dễ thấy nhưng quan trọng nhất mà TensorFlow cung cấp cho việc lập trình Machine Learning chính là abstraction. Thay vì phải đối phó với những tình huống rườm rà từ việc thực hiện triển khai các thuật toán, hay tìm ra cách hợp lý để chuyển output của 1 chức năng sang input của 1 chức năng khác, giờ đây bạn có thể tập trung vào phần logic tổng thể của 1 ứng dụng hơn. TensorFlow sẽ chăm sóc phần còn lại thay cho bạn.

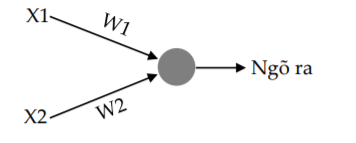
Ngoài ra TensorFlow còn cung cấp các tiện ích bổ sung cho các lập trình viên cần debug cũng như giúp bạn tự suy xét các ứng dụng TensorFlow. Chế độ Eager execution cho phép bạn đánh giá và sửa đổi từng operation của biểu đồ 1 cách riêng biệt và minh bạch, thay vì phải dựng toàn bộ biểu đồ dưới dạng 1 đối tượng độc lập vốn khá mơ hồ hay phải đánh giá chung tổng thể. Cuối cùng, 1 tính năng khá độc đáo của TensorFlow là TensorBoard. TensorBoard cho phép bạn quan sát 1 cách trực quan những gì TensorFlow đang làm.

TensorFlow còn có nhiều cải tiến từ sự hậu thuẫn từ các ekíp thương mại hạng A tại Google. Google không những tiếp lửa cho tiến độ nhanh chóng cho sự phát triển đằng sau dự án, mà còn tạo ra nhiều phục vụ độc đáo xung quanh TensorFlow để nó dễ dàng deploy và sử dụng: như silicon TPU mình đã nói ở trên để tăng tốc hiệu suất đám mây Google, 1 online hub cho việc chia sẻ các model được tạo với framework, sự hiện diện của in-browser và gần gũi với mobile của framework, và nhiều hơn thế nữa…

*Lưu ý:* Trong 1 số công việc training, vài chi tiết về việc triển khai của TensorFlow làm cho nó khó có thể quyết định được hoàn toàn kết quả training model . Đôi khi 1 model được train trên 1 hệ thống này sẽ có thay đổi 1 chút so với 1 model được train trên hệ thống khác, ngay cả khi chúng được cung cấp dữ liệu như nhau. Các nguyên nhân cho điều này cũng xê xích hay một số hành vi khi không được xác định khi sử dụng GPU. Điều này nói rằng, các vấn đề đó có thể giải quyết được, và đội ngũ của TensorFlow cũng đang xem xét việc kiểm soát nhiều hơn để ảnh hưởng đến tính quyết định trong quy trình làm việc.

# II. Nhận dạng chữ số viết tay dùng mạng neuron nhân tạo

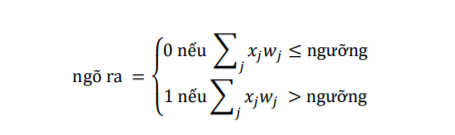
## 2.1 Neuron nhân tạo

Neuron nhân tạo được xây dựng trên ý tưởng từ neuron sinh học, đơn vị cấu tạo thành hệ thần kinh của não bộ con người. Những neuron này có nhiệm vụ nhận tính hiệu từ não bộ và truyền tín hiệu đến các bộ phận khác nhau trên cơ thể. Một cách tương tự, ta có các neuron nhân tạo hay perceptron là đơn vị cấu thành mạng neuron nhân tạo. Một perceptron sẽ nhận các ngõ vào có giá trị x1, x2, ... xn và ngõ ra của perceptron sẽ cho ra giá trị nhị phân. Hình 1 mô tả cấu trúc của một perceptron đơn giản với 2 giá trị đầu vào X1, X2.Mô hình Perceptron đơn giản.

Hình 1: Mô hình Perceptron đơn giản.

Nhiệm vụ của perceptron là sử dụng các giá trị ngõ vào để tính toán và xác định giá trị ngõ ra là 0 hoặc 1. Trong một mạng neuron nhân tạo, ngõ ra của neuron lớp trước sẽ là đầu vào của lớp tiếp theo.

### 2.1.1 Trọng số (Weight)

Weight là con số biểu thị mức độ quan trọng của ngõ vào so với ngõ ra. Giá trị ngõ ra của perceptron phụ thuộc vào tổng giữa trọng số và ngõ vào:

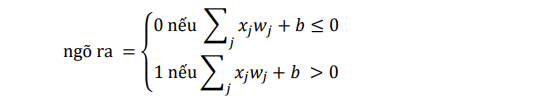
Trong đó :

xj : ngõ vào thứ j của perceptron; wj:trọng số của ngõ vào xj ; mức ngưỡng quyết định giá trị ngõ ra. Hàm xác định ngõ ra ở (1) còn được gọi là hàm step

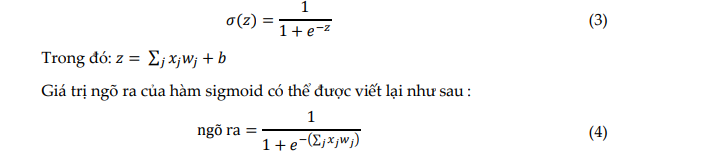
### 2.1.2 Bias

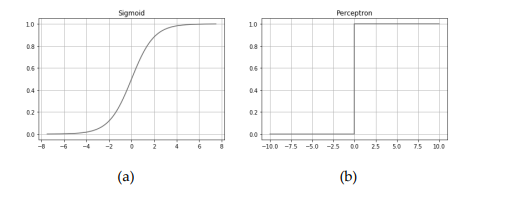
Để đơn giản cho perceptron quyết định giá trị đầu ra 0 hay 1, có thể thay thế giá trị ngưỡng (threshold) bằng nghịch đảo số Bias hay ngưỡng = -b.

Và có thể viết lại như sau:



### 2.1.3 Neuron sigmoid và hàm sigmoid

Tương tự với perceptron, neuron sigmoid có nhiều ngõ vào (x1, x2, x3<) tương ứng có các trọng số cho mỗi đầu vào khác nhau (w1, w2, w3<) và bias b. Nhưng thay vì chỉ nhận một trong hai giá trị 0/1, chúng có thể có bất kì giá trị thực nào giữa 0 và 1 (ví dụ: 0,45), ngõ ra được xác định bởi hàm kích hoạt

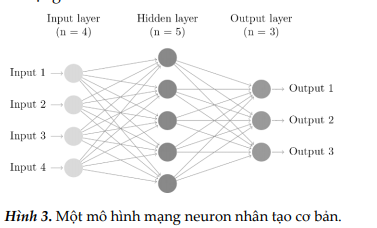
Hàm sigmoid, dùng để tính toán ngõ ra của các neuron giữa các lớp của mạng,cho phép giá trị ngõ ra là các số thực nằm trong khoảng 0 đến 1, thay vì chỉ nhận 2 giátrị 0/1 đơn thuần như ở hàm step. Từ đó, giúp cho quá trình học của neuron linh hoạt hơn, các tham số được hiệu chỉnh tốt hơn thay vì thay đổi đột ngột. Quá trình trên được biểu diễn thông qua đồ thị của 2 hàm tương ứng, mô

## 2.2. Mô hình mạng Neuron nhân tạo

### 2.2.1 Cấu trúc mạng neuron nhân tạo

Về cơ bản, Multi-layer Perceptron (MLP) có tối thiểu 3 lớp: một lớp ngõ vào, một lớp ngõ ra và một hoặc nhiều lớp ẩn.Trong đó, lớp ngõ vào tiếp nhận dữ liệu vào mạng neuron; lớp ẩn (hidden layer) dùng để tính toán các thông số dựa trên dữ liệu từ lớp ngõ vào và chuyển tiếp kết quả đến lớp tiếp theo; lớp ngõ ra thực hiện định gía và đưa ra kết quả dựa trên các thuật toán thích hợp.

Các mô hình mạng MLP sử dụng ngõ ra lớp trước làm ngõ vào lớp sau, không có sự phản hồi từ lớp ngõ ra về lại lớp ngõ vào được gọi là mạng feed-forward và sự liên kết các neuron giữa các lớp được gọi là fully connected [3] hay mạng kết nối hoàn toàn. Mạng fully connected có ưu điểm là đơn giản, dễ triển khai tuy nhiên nhược điểm là số lượng thông số và số lượng phép tính trong mạng rất lớn. Ví dụ, với mô hình mạng như hình 3, bỏ qua thông số bias, số lượng trọng số đã là: 4 x 5 + 5 x 3 = 35 trọng số.

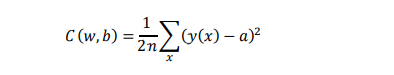


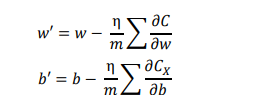
### 2.2.2 Phương pháp “học” trong mạng MLP

Hiệu suất của mạng được đánh giá bằng cách sử dụng hàm tổn hao hay cost function . Dựa vào hàm này để định giá một mô hình mạng “học” tốt hay không tốt. Giá trị của hàm tổn hao có giá trị lớn nếu ngõ ra dự đoán của mạng khác với nhãn thực sự vị ngược lại

Trong đó:

*C(w,b)*: hàm tổn hao; n:tổng số nhãn; a: ngõ ra của mạng dự đoán; y(x): giá trị ngõ ra thực sự tương ứng với ngõ vào x.



Việc huấn luyện mạng neuron nhân tạo là tìm ra các giá trị phù hợp của trọng số và bias sau mỗi vòng lặp từ những giá trị ngẫu nhiên được khởi tạo ban đầu để có được ngõ ra như mong muốn. Bằng phương pháp Gradient Descent cho phép thay đổi các trọng số và bias bằng cách lấy đạo hàm của hàm tổn hao. Cụ thể, các tham số sẽđược cập nhật như sau:

Trong đó : tương ứng w’, b’: là các trọng số và bias mới của neuron được cập nhật; tỷ lệ học; m: mini-batch size được tạo ra bằng cách chia nhỏ số lượng dữ liệu. Các thông số này sẽ được khảo sát rõ ở phần 4.

# III. Chương trình nhận dạng chữ số và phép tính cơ bản

## 3.1 Mô tả hoạt động các hàm

Mô tả hoạt động của các hàm trong code

Hàm build model:

Text

Description automatically generated

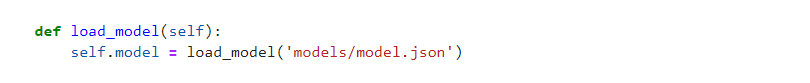
Hàm này dùng để xây dựng một model cho quá trình huấn luyện, ở đây chúng em chỉ sử dụng các tầng Dense layer và giảm số node của input qua mỗi tầng và tầng cuối cùng là 10 node vì output của bài toán này có 10 classes.

Về hàm loss, chúng em sử dụng SparseCategoricalCrossentropy vì hàm loss này thường được dùng cho bài toán Multiclass-Classification với label chưa được one-hot encoded.

Optimizer chúng em sử dụng Adam với learning\_rate = 10-3

Cuối cùng là hàm compile với các thông số được khởi tạo ở trên.

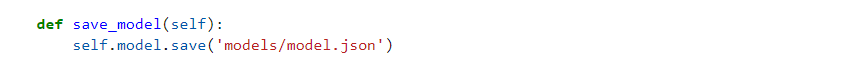
Hàm load model:



Hàm này được sử dụng để load model sau khi model đã được lưu để predict kết quả đầu ra có phù hợp với đầu vào hay không.

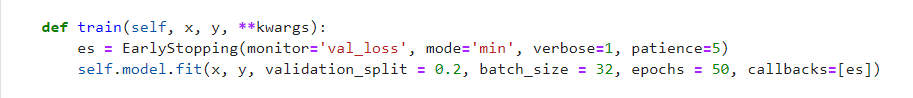


Hàm save model:



Hàm này được sử dụng để lưu model sau khi huấn luyện vào thư mục “models” để sử dụng cho việc load model lên và predict.

Hàm train model:



Hàm train dùng để huấn luyện model, hàm này có sử dụng hàm EarlyStoping() để tránh trường hợp bị overfit khi huấn luyện, với patience=5 có nghĩa là nếu trong 5 epoch liên tiếp mà accuracy không tăng thì quá trình huấn luyện sẽ dừng lại.

Hàm train với x và y là dữ liệu huấn luyện, validation\_split=0.2 có nghĩa là lấy 20% dữ liệu trong tập dữ liệu ban đầu để làm tập valid, batch\_size=32 là số lượng sample sẽ được truyền qua mạng, epoch là một lần vượt qua toàn bộ tập dữ liệu.

Hàm predict:

A picture containing scatter chart

Description automatically generated

Hàm này nhận một dữ liệu thử nghiệm để predict kết quả đầu ra có chính xác hay không.

## 3.2. Hướng dẫn sử dụng

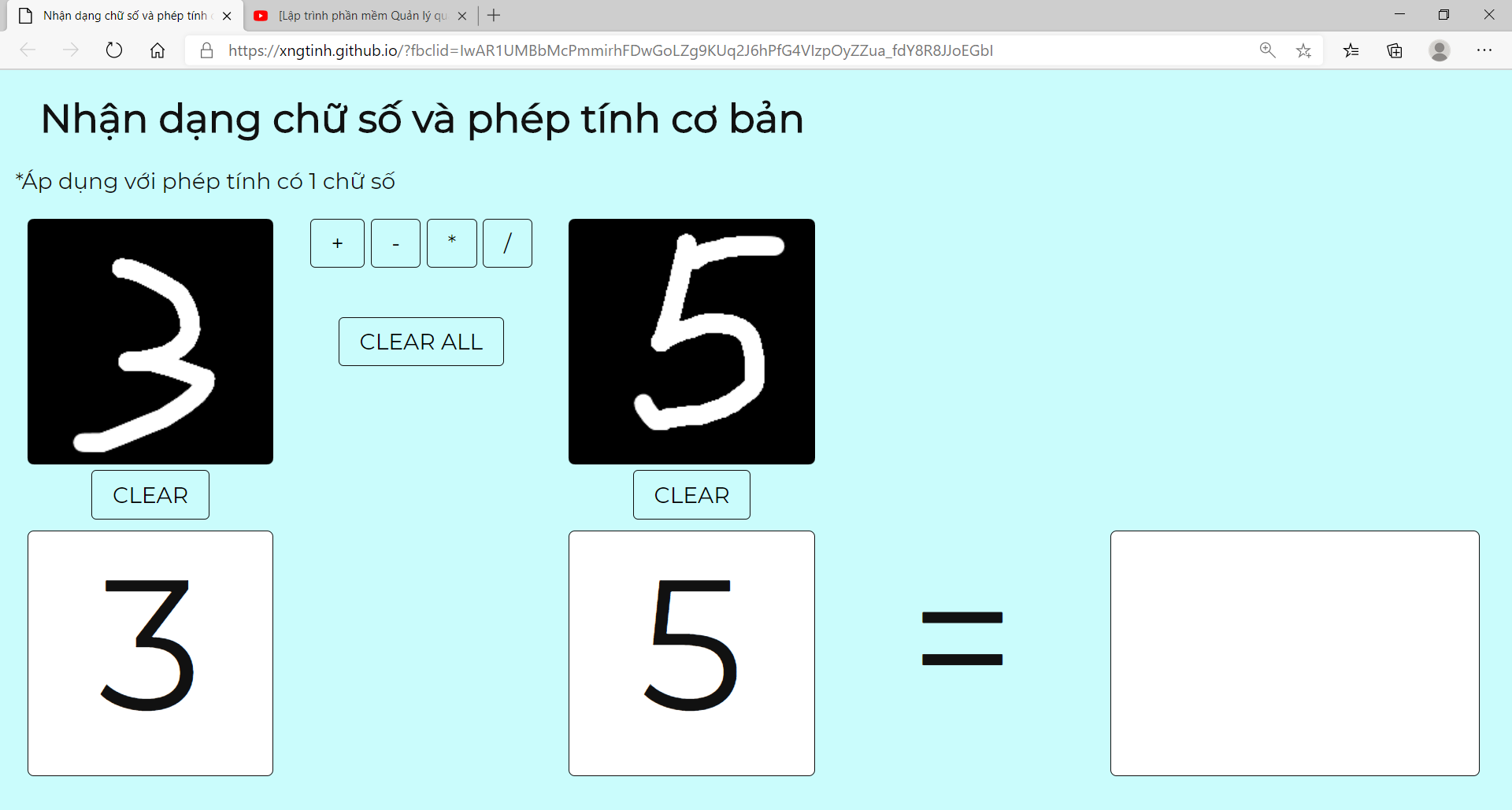
Khi vào trang <https://xngtinh.github.io/>, chúng ta sẽ có màn hình chính và các bước sử dụng như sau:



**Bước 1: Vẽ giá trị vào một trong 2 ô màu đen bất kì.**

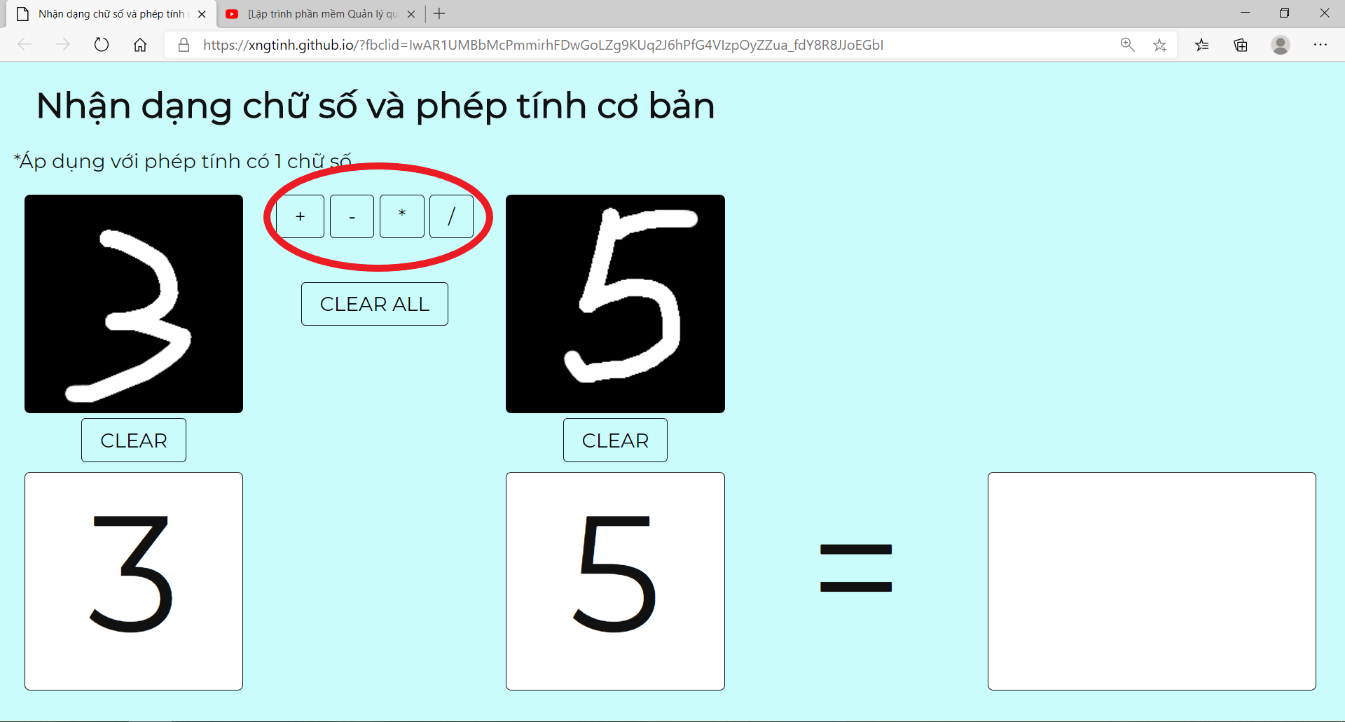


**Bước 2: Vẽ giá trị vào ô màu đen còn lại.**

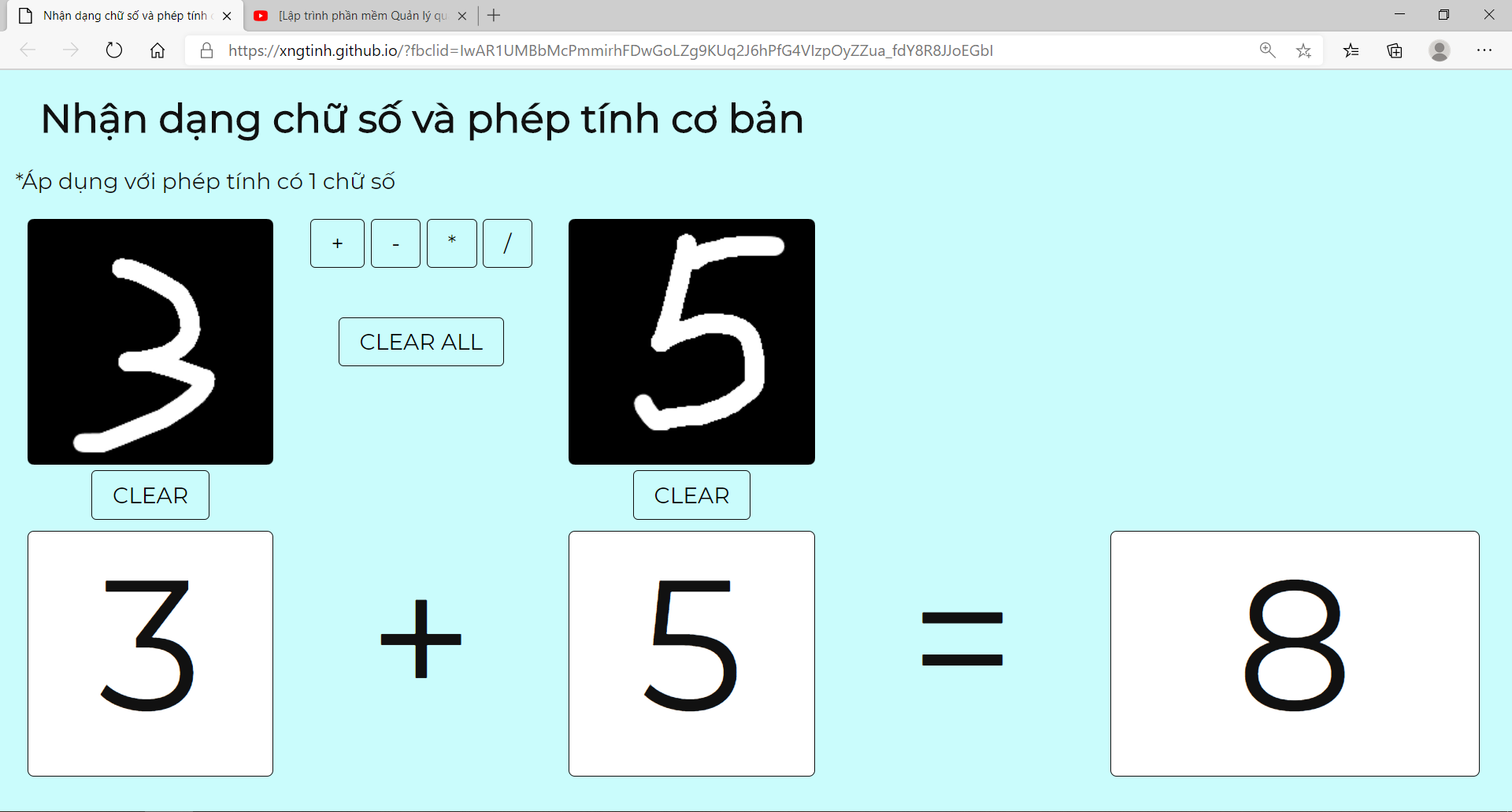


*\*Lưu ý : vẽ số vào 2 ô màu đen thì mới được thực hiện các bước tiếp theo*

**Bước 3 : Chọn 1 trong số các phép tính.**



**Bước 4: Nhận kết quả.**



Lưu ý trong quá trình nhập liệu nếu có sai sót thì ta có thể nhập lại bằng cách nhấn nút CLEAR và nhập lại tất cả bằng nút CLEAR ALL.

|  |  |
| --- | --- |
| **ĐÁNH GIÁ CỦA GIẢNG VIÊN** | |
|  |  |
|  |  |