Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh

TRUNG TÂM TIN HỌC

Natural Language Processing with Deep Learning

Bài 3: **PYTORCH** & DEEP LEARNING MODELS

https://csc.edu.vn/data-science-machine-learning/natural-language-processing-with-deep-learning_293

2023

PYTORCH & DEEP LEARNING MODELS

I. Tổng quan về các mô hình Deep Learning

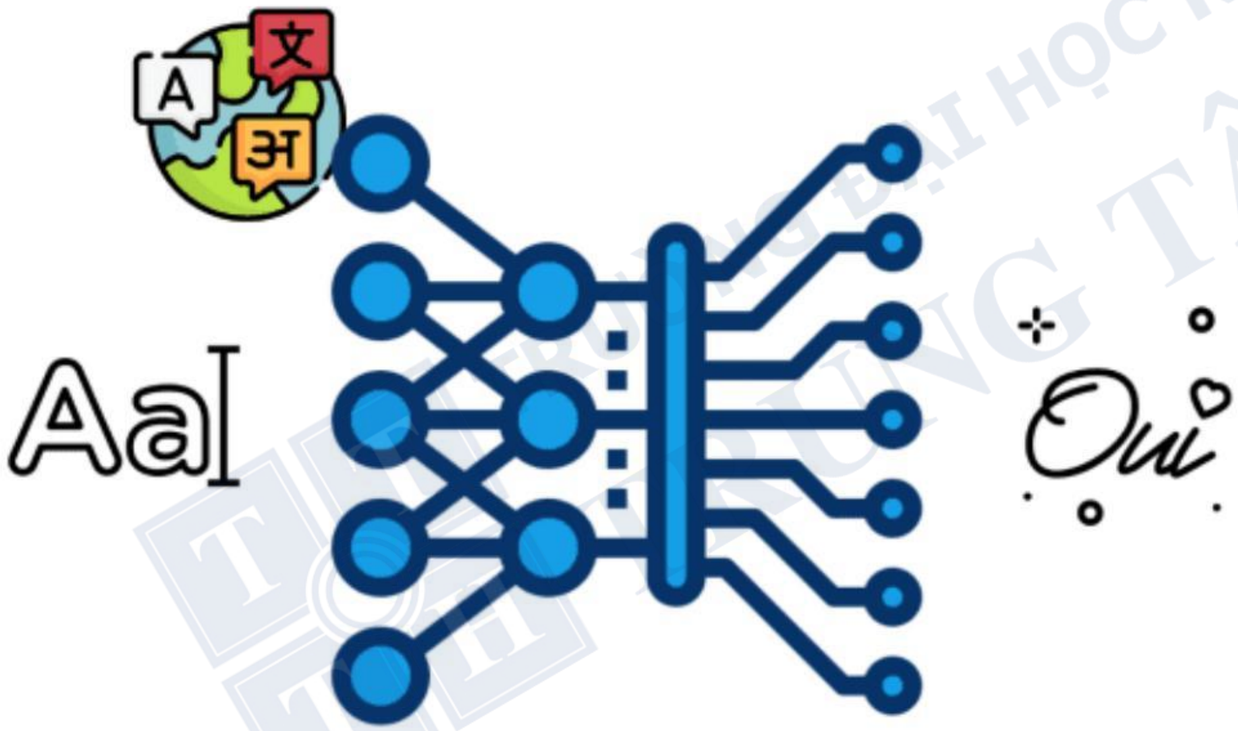
II. Mô hình Linear và Logistic Regression

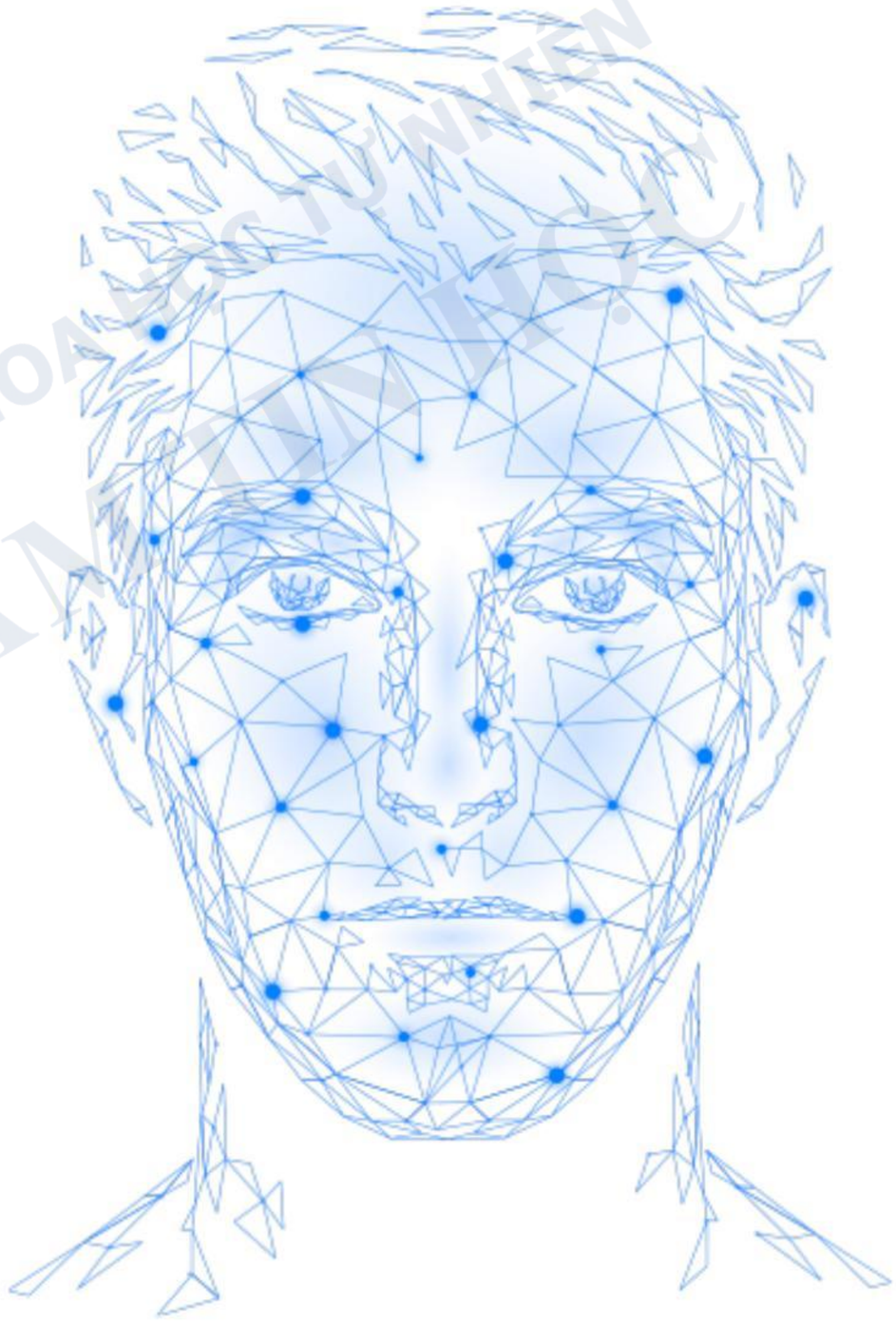
III. Mô hình mạng Neural Network

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRUNG TÂM TIN HỌC

Tổng quan Deep Learning

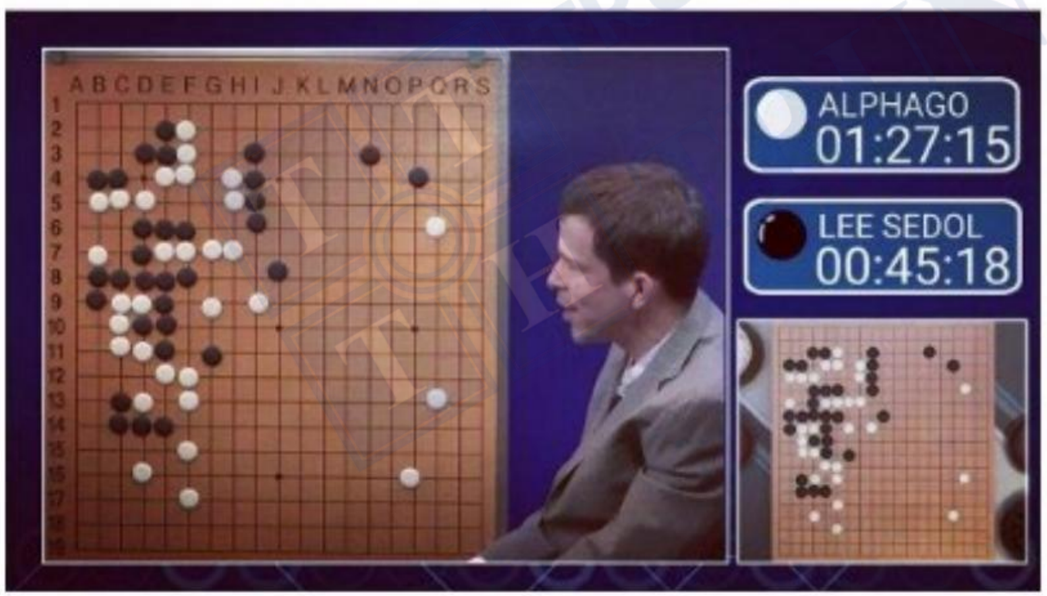
- **Deep Learning** là nhánh quan trọng trong Machine Learning, áp dụng rộng rãi trong tác vụ **xử lý hình ảnh**





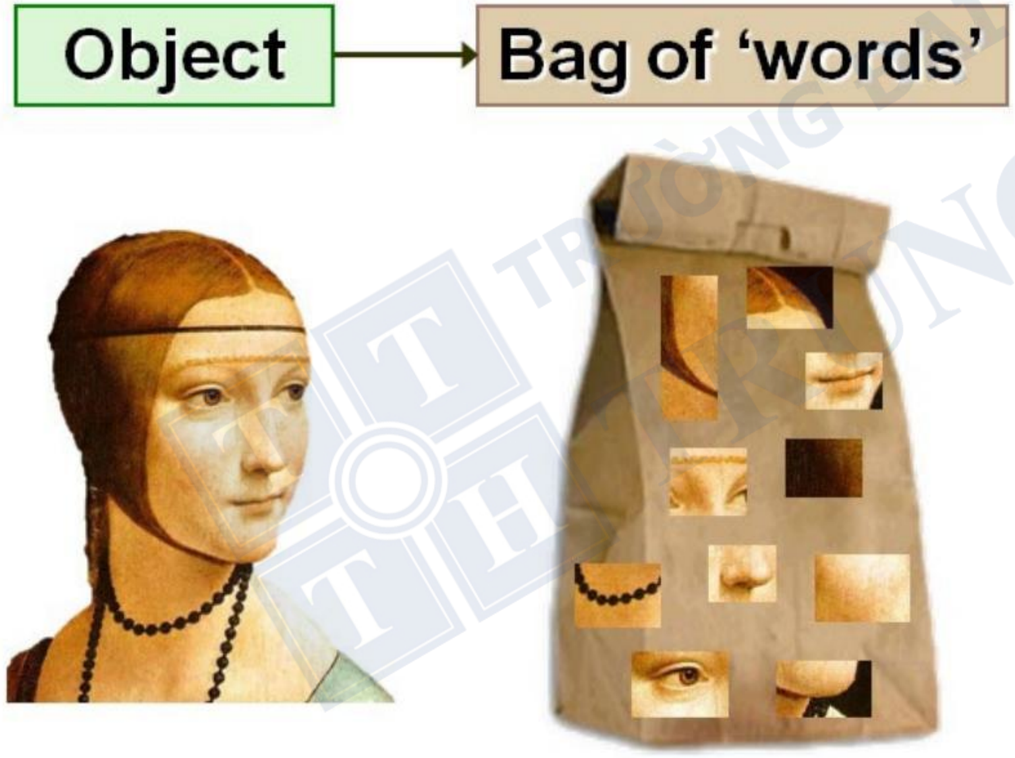
hay **xử lý ngôn ngữ tự nhiên**.

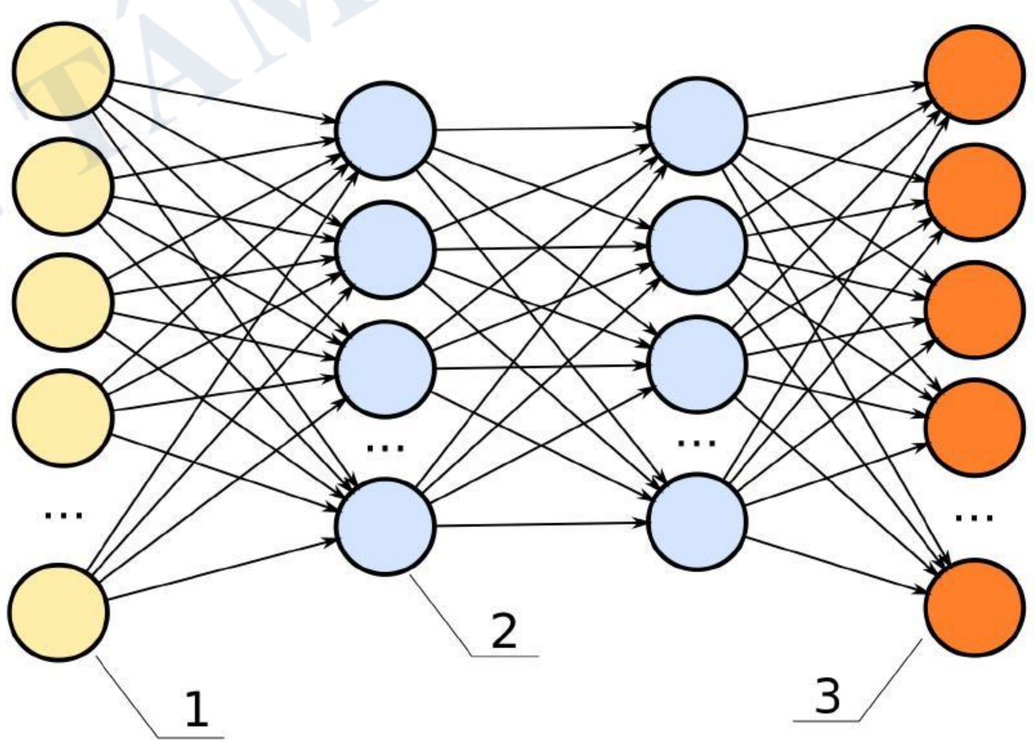
Tổng quan Deep Learning



Tổng quan Deep Learning

- Deep Learning dùng mạng **neural network** để học và hiểu các đặc trưng phức tạp từ dữ liệu input.
- Mạng này có khả năng tự học và cải thiện theo thời gian, giúp nó linh hoạt khi giải quyết các bài toán phức tạp.





PYTORCH & DEEP LEARNING MODELS

I. Tổng quan về các mô hình Deep Learning

II. Mô hình Linear và Logistic Regression

III. Mô hình mạng Neural Network

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRUNG TÂM TIN HỌC

Tổng quan Loss function

Các bước tính **Loss Function**

1. Khởi tạo mạng **neural network** với trọng số ngẫu nhiên.
2. Thực hiện **forward pass**.
3. Tính **hàm loss**.
4. Tính toán **gradient**.
5. Thay đổi **trọng số** dựa trên gradient.

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRUNG TÂM TIN HỌC

Tổng quan Loss function

Loss Function cho deep learning models

- ✓ Đối với Regression: **Mean Square Error**.
- ✓ Đối với Classification: **Cross-Entropy Loss**.
- ✓ Đối với các vấn đề phức tạp hơn (như object detection): các hàm loss phức tạp hơn như **Softmax Cross-Entropy Loss.**

Softmax Cross-Entropy Loss

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

cat	3.2
car	5.1
frog	-1.7

datacamp

ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRƯỜNG ĐẠI HỌC
TRUNG TÂM TIN HỌC

Softmax Cross-Entropy Loss

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

cat	3.2	exp →	24.5
car	5.1	exp →	164.0
frog	-1.7	exp →	0.18

Probabilities must be >= 0

unnormalized probabilities

datacamp

ĐẠI HỌC KHOA HỌC TỰ NHIÊN TRƯỜNG ĐẠI TRUNG TÂM TIN HỌC

Softmax Cross-Entropy Loss

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

		exp	Probabilities must be >= 0	normalize	Probabilities must sum to 1
cat	**3.2**		**24.5**		**0.13**
car	5.1	→	164.0	→	0.87
frog	-1.7		0.18		0.00
			unnormalized probabilities		probabilities

datacamp

Softmax Cross-Entropy Loss

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

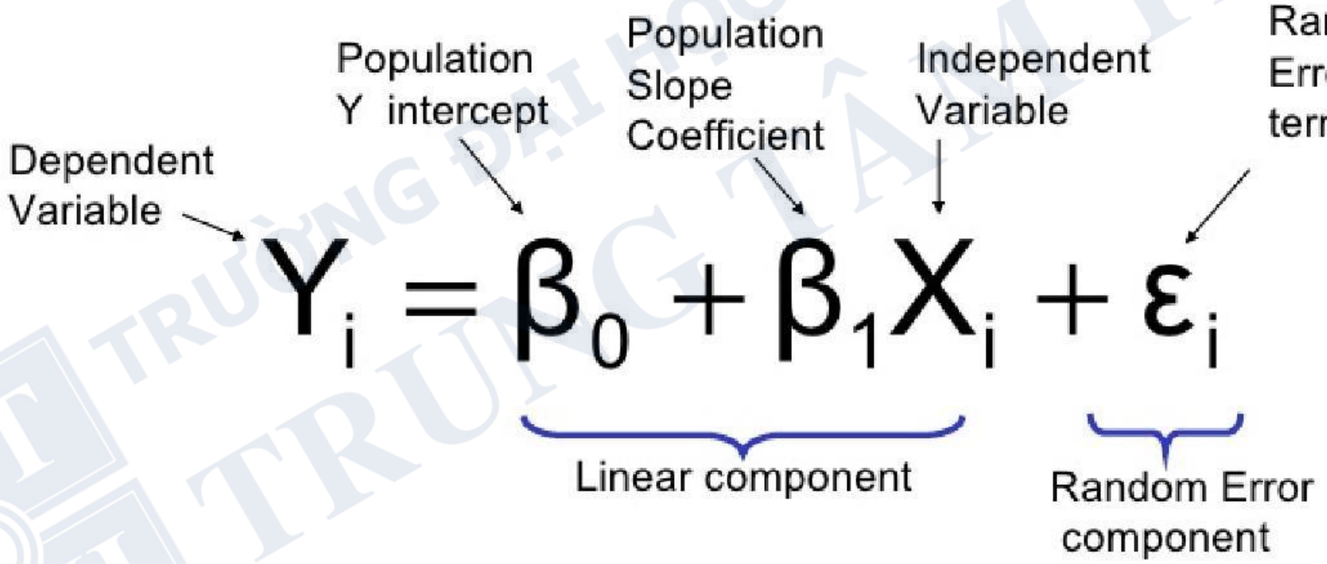
		exp	Probabilities must be >= 0	normalize	Probabilities must sum to 1
cat	3.2	→	24.5	→	0.13
car	5.1	→	164.0	→	0.87
frog	-1.7	→	0.18	→	0.00
			unnormalized probabilities		probabilities

L = - ln(0.13) = 2.0404

datacamp

Tổng quan Linear Regression

- **Linear Regression** là mô hình dự đoán giá trị output dựa trên các input và mối quan hệ tuyến tính giữa chúng.
- Công thức toán học của Linear Regression là:



Ứng dụng Linear Regression

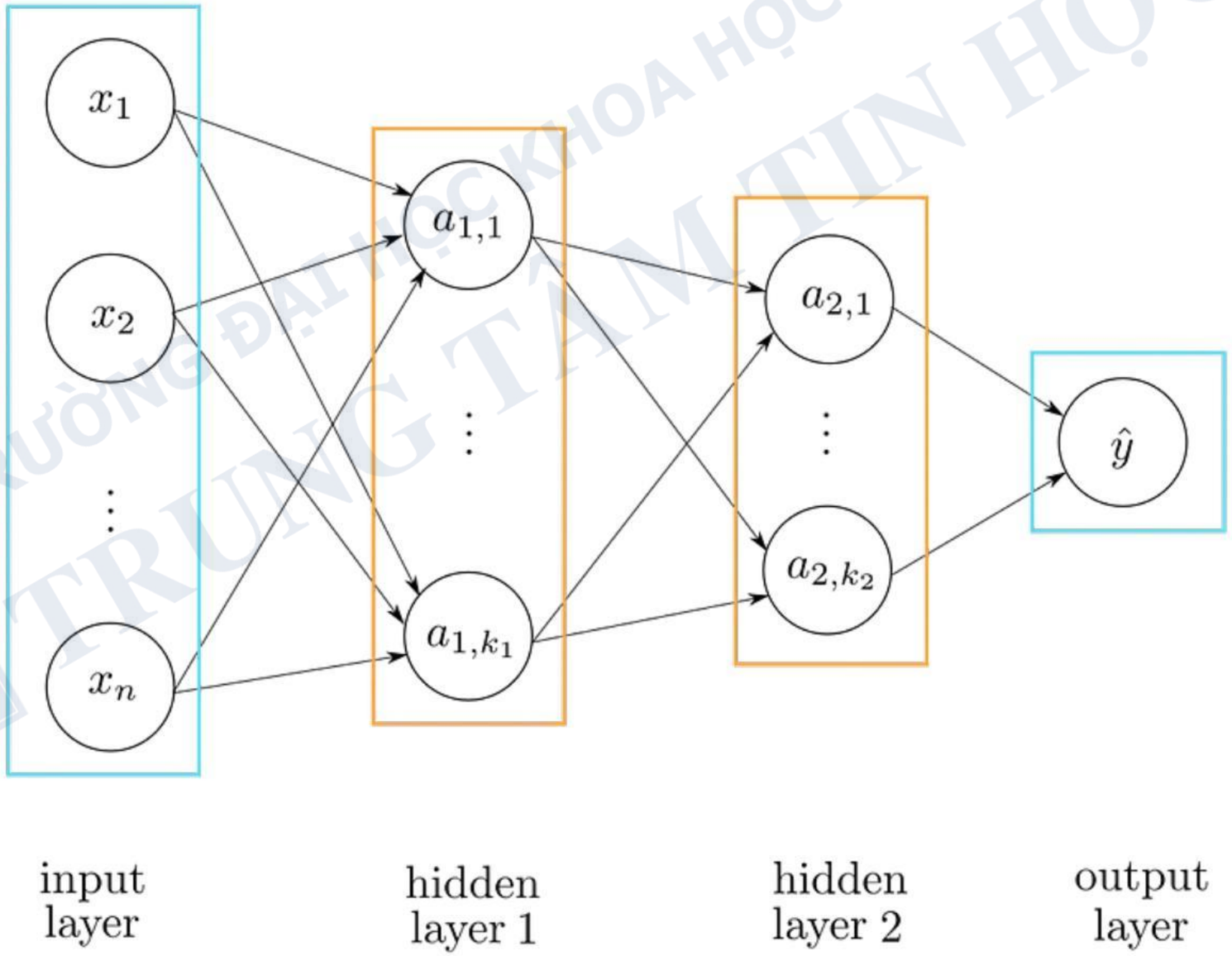
- **Linear Regression** có nhiều ứng dụng trong thực tế như dự đoán giá nhà dựa trên diện tích, dự đoán doanh thu của một công ty dựa trên các chỉ số kinh doanh, và dự đoán xu hướng thị trường dựa trên các yếu tố kinh tế...

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRUNG TÂM TIN HỌC

Neural Network & Linear Regression

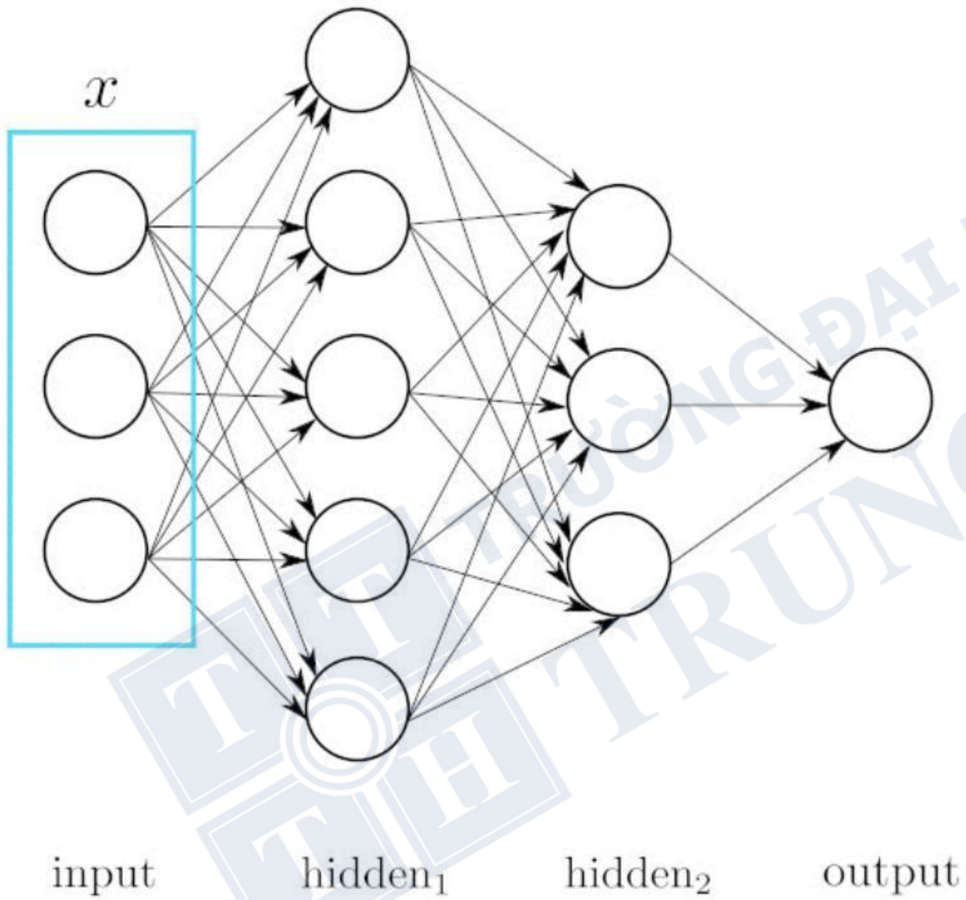
Forward Pass

$$x \xrightarrow{f_1} f_1(W_1 x + b_1) \xrightarrow{f_2} f_2(W_2 a_1 + b_2) \xrightarrow{\Sigma} \hat{y}$$



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRUNG TÂM TIN HỌC

Ví dụ Neural Network & Linear Regression

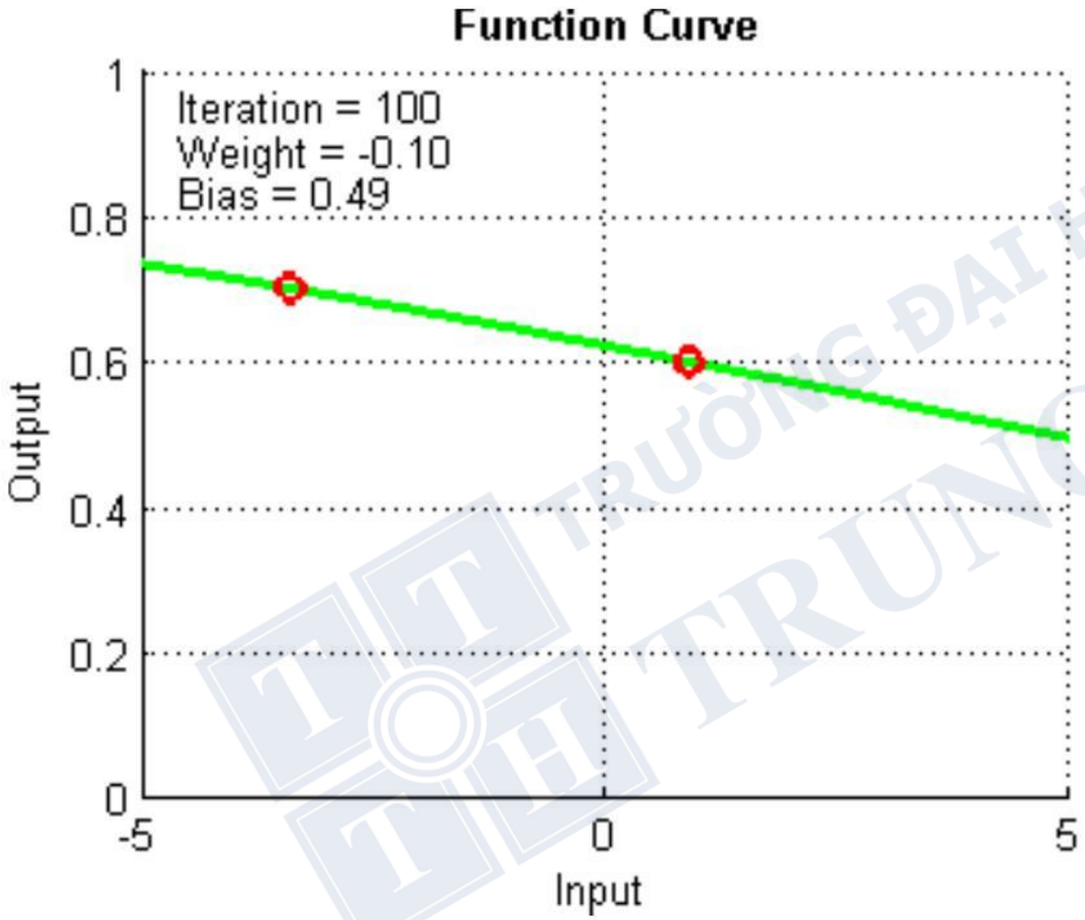


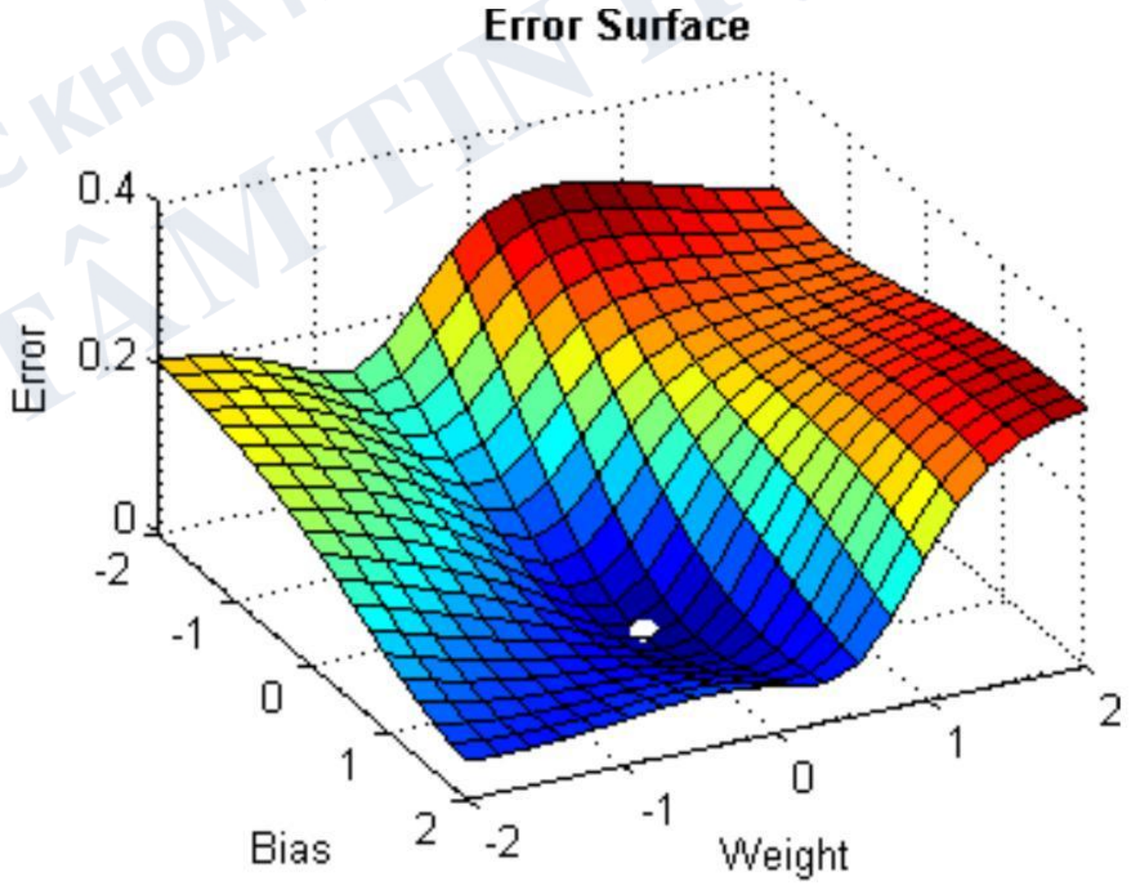
Start with the input layer:

ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRUNG TÂM TIN HỌC

Bài toán Linear Regression

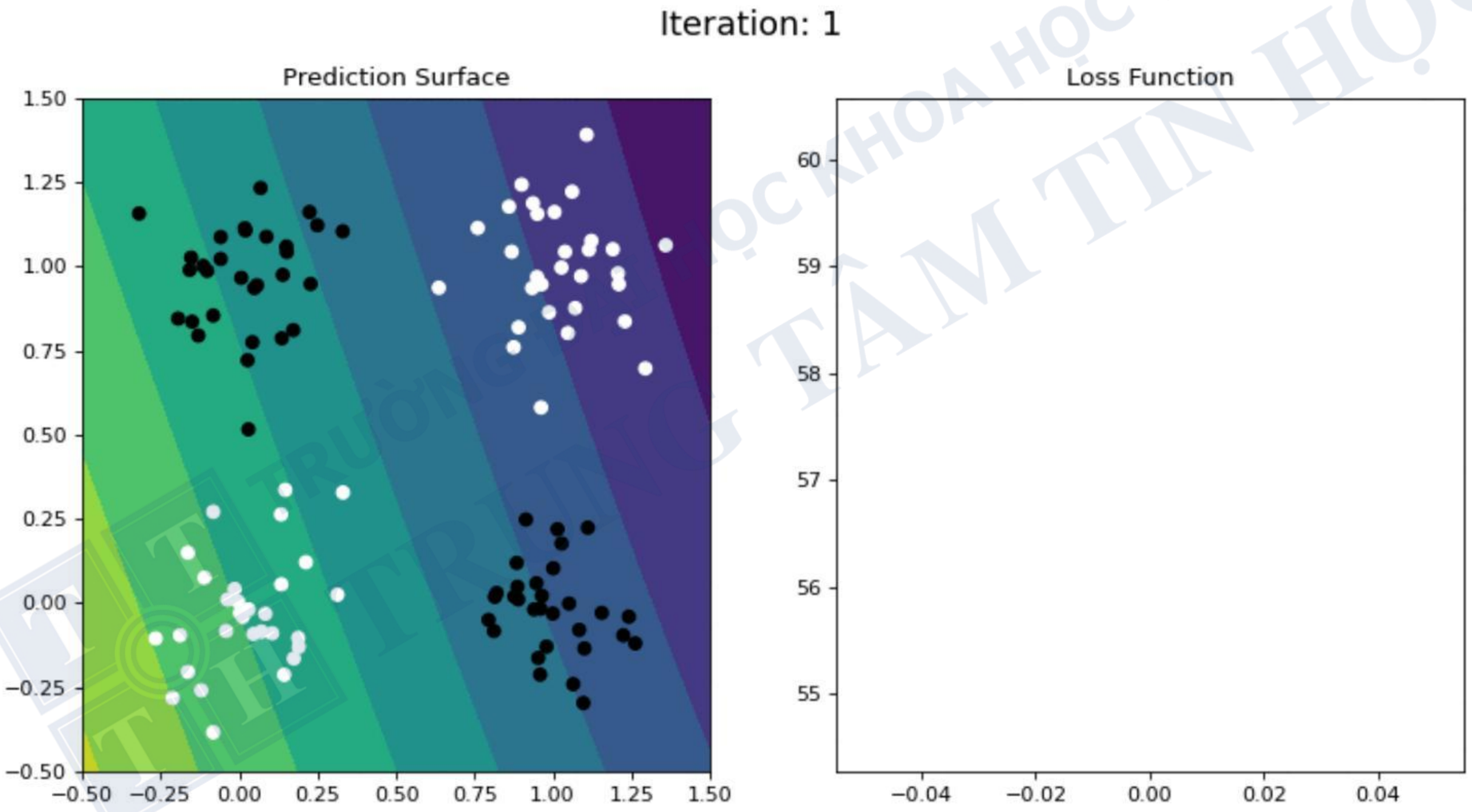
Loss Function bài toán **non-linear**





Bài toán Non-linear Regression

Loss Function bài toán **non-linear**



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRUNG TÂM TIN HỌC

PYTORCH & DEEP LEARNING MODELS

I. Tổng quan về các mô hình Deep Learning

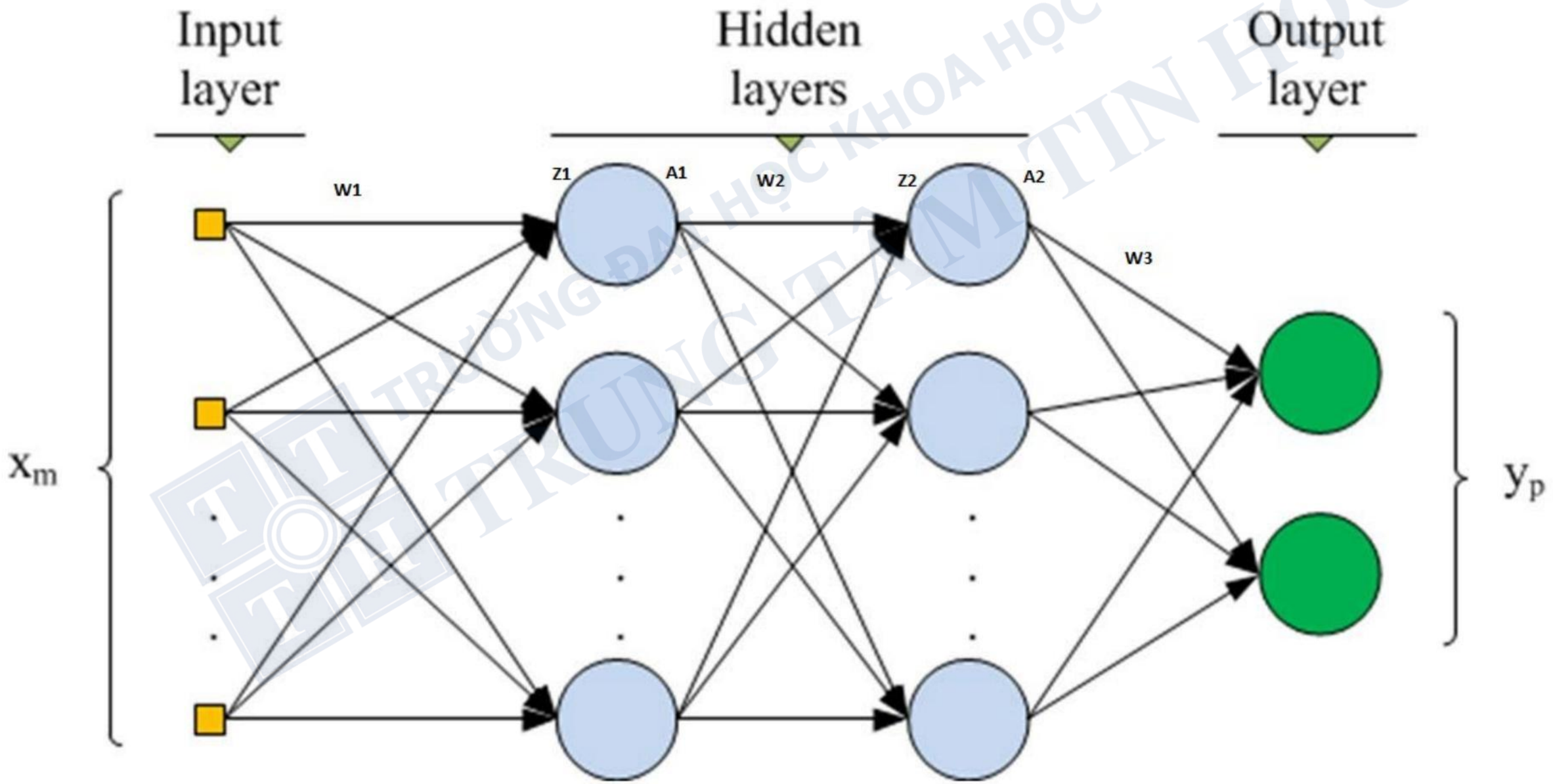
II. Mô hình Linear và Logistic Regression

III. Mô hình mạng Neural Network

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRUNG TÂM TIN HỌC

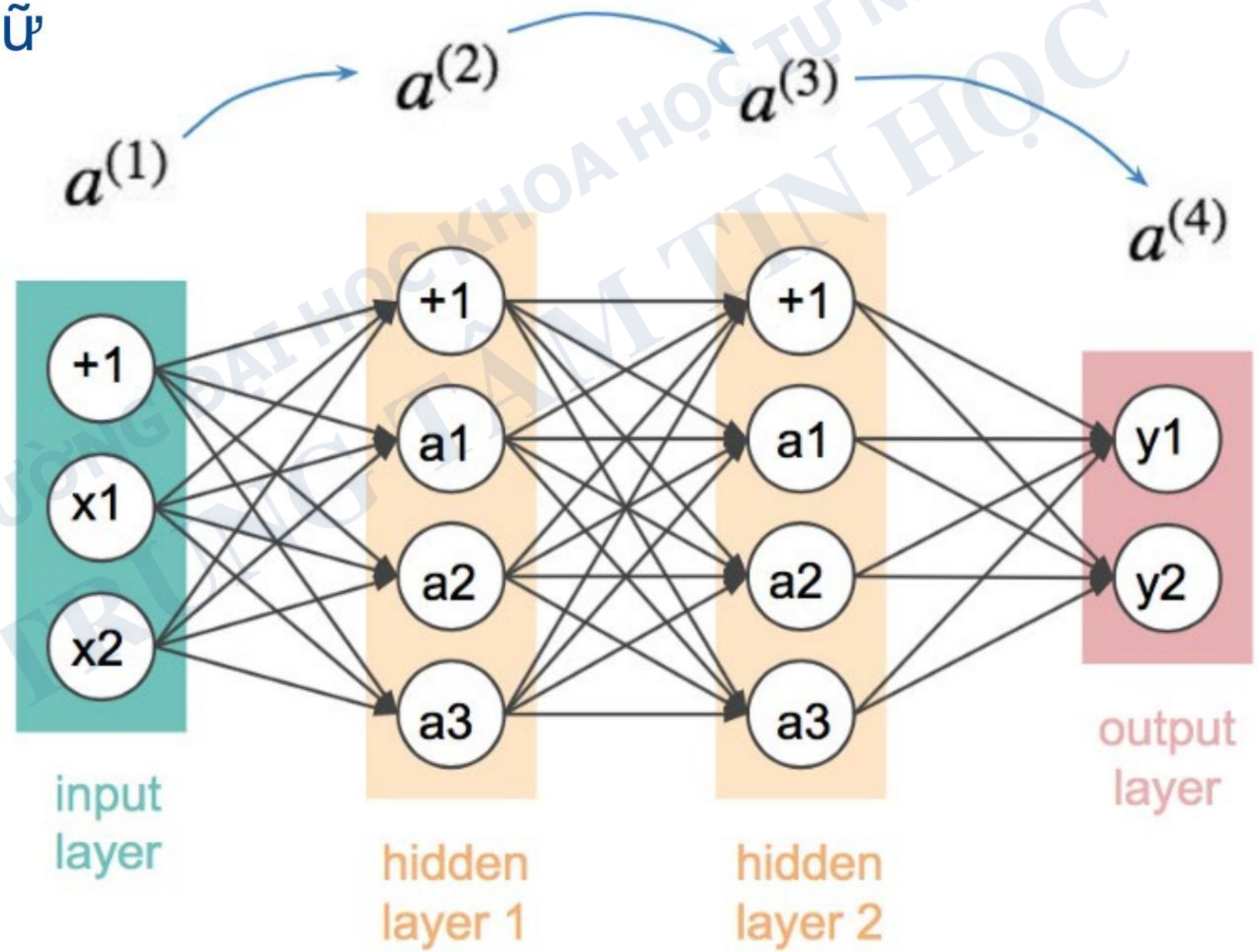
Tổng quan Neural Network

- Mạng **fully connected neural network** cơ bản:



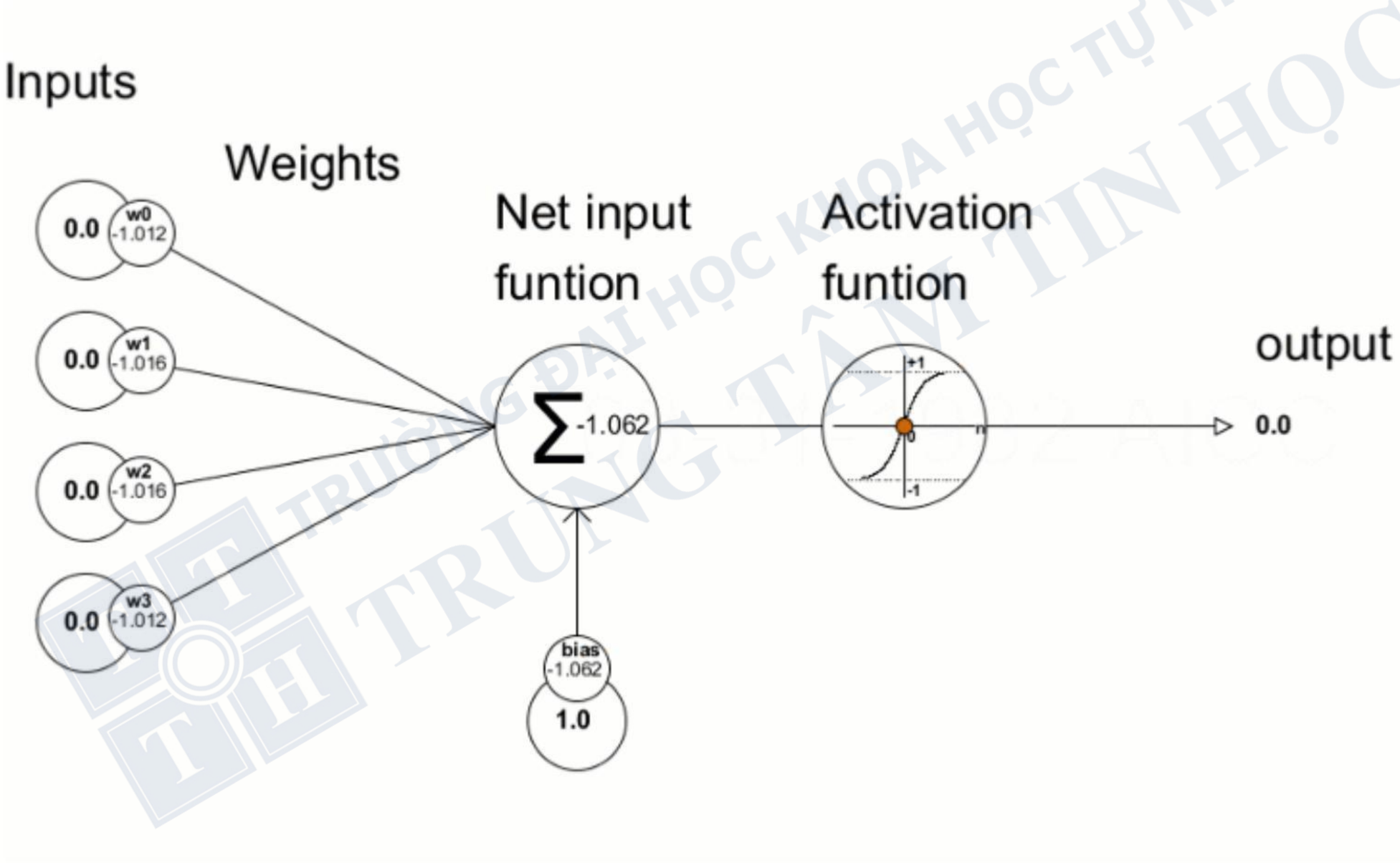
Forward Propagation với Pytorch

- Là quá trình tính toán và truyền dữ liệu từ input đến output trong mạng **neural network**.



Tổng quan Forward Propagation

- Sơ đồ tính toán theo **forward propagation**



Forward Propagation với Pytorch

```
import torch
import torch.nn as nn

class NeuralNetwork(nn.Module):
  def __init__(self):
    super(NeuralNetwork, self).__init__()

    self.fc1 = nn.Linear(10, 5)

    self.fc2 = nn.Linear(5, 2)
  def forward(self, x):

    x = torch.relu(self.fc1(x))

    x = self.fc2(x)
    return x

model = NeuralNetwork()

input_tensor = torch.randn(1, 10)

output_tensor = model(input_tensor)
print(output_tensor)
```

Định nghĩa mạng **neural network** với **2 lớp fully connected**

Lớp thứ nhất có input size = 10 & output size = 5, lớp thứ hai có input size = 5, output size = 2

Dùng hàm kích hoạt **ReLU** cho output lớp fc1 và không dùng cho fc2

Tạo một instance của mạng neural network

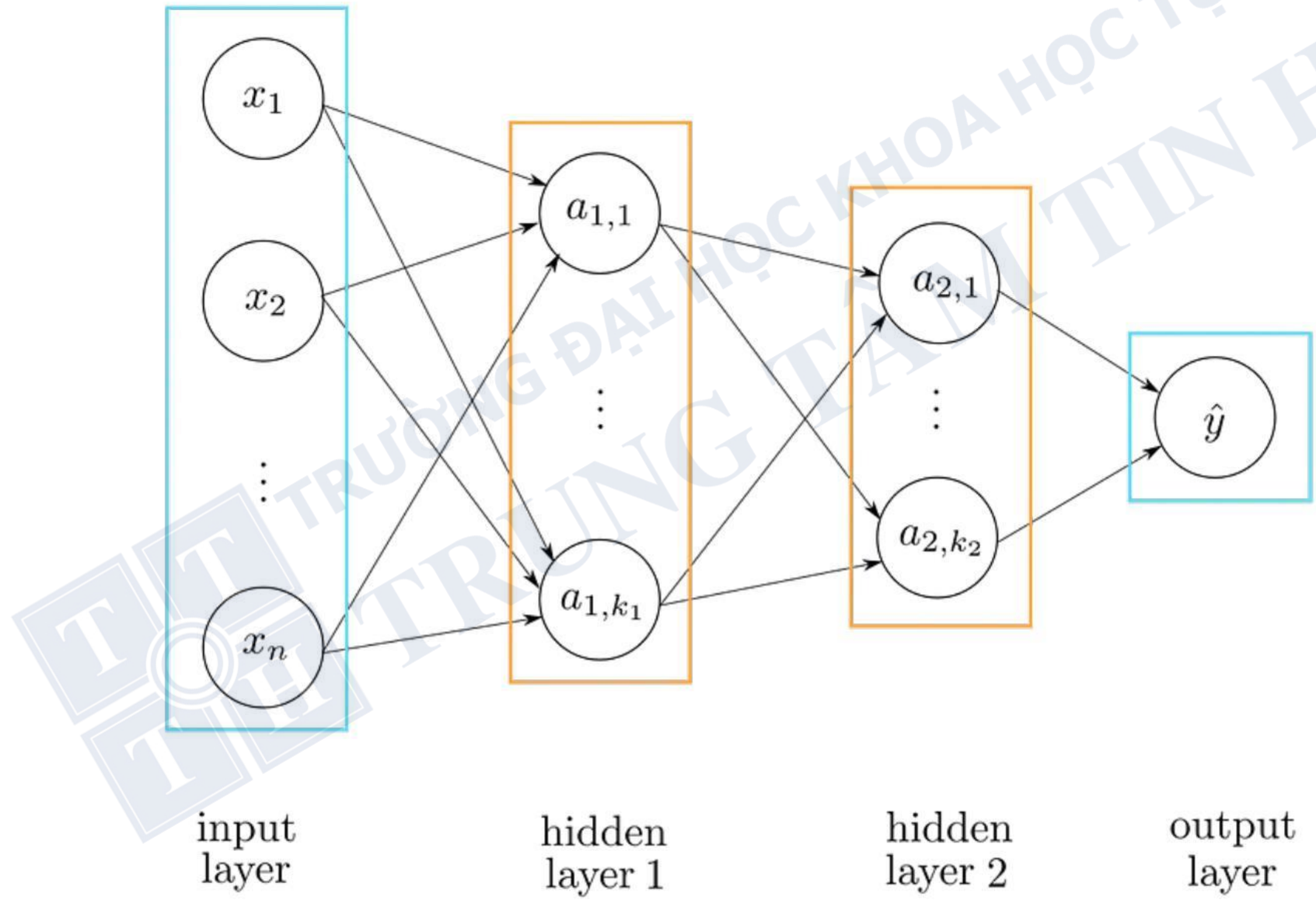
Tạo một tensor đại diện cho input

Thực hiện **forward propagation**

```
tensor([[0.1234, -0.5678]])
```

Neural Network Forward Pass

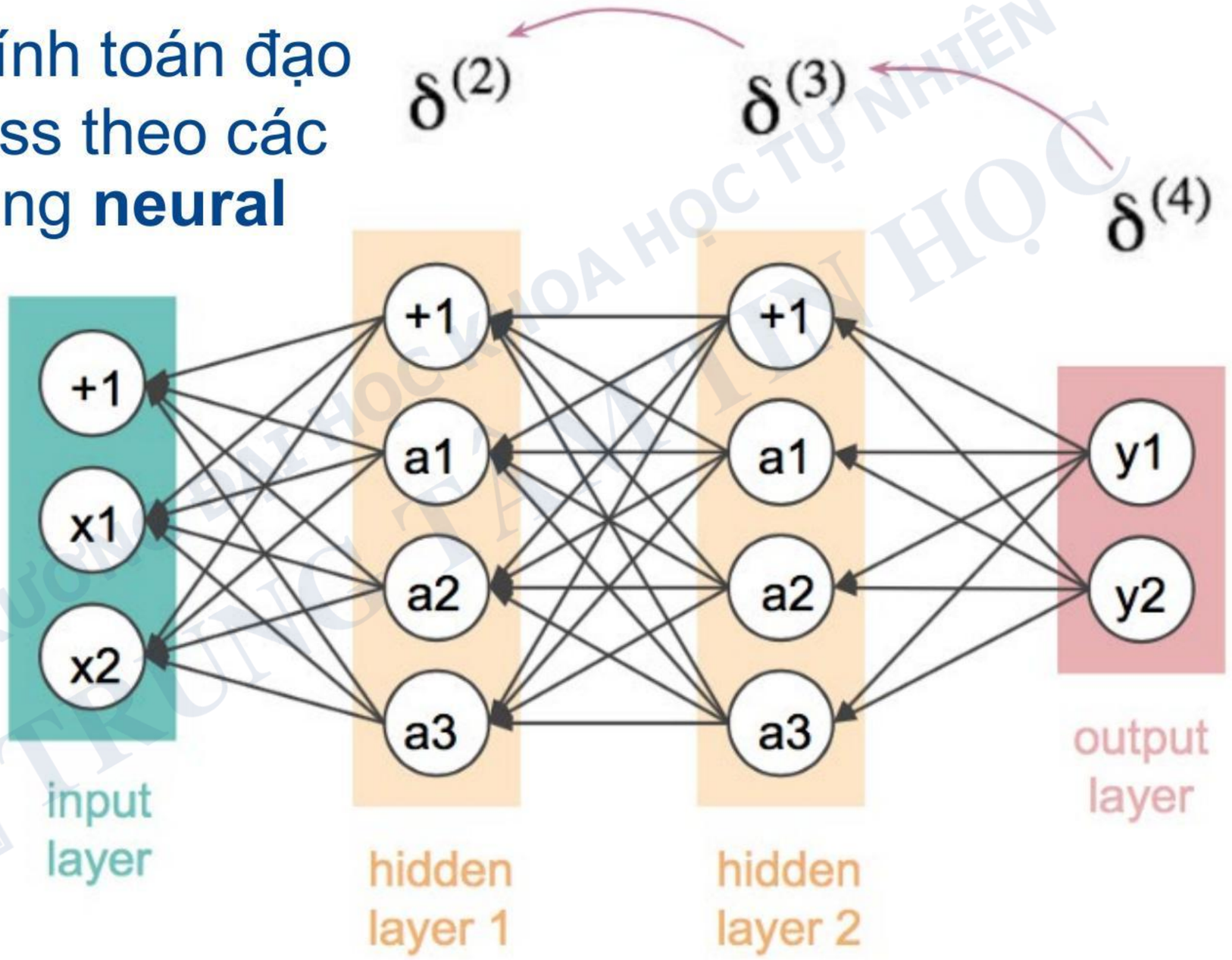
$$x \xmapsto{f_1} f_1(W_1 x + b_1) \xmapsto{f_2} f_2(W_2 a_1 + b_2) \xmapsto{\sum} \hat{y}$$



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TRUNG TÂM TIN HỌC

Tổng quan Back Propagation

- Là quá trình tính toán đạo hàm của hàm loss theo các trọng số của mạng **neural network** để tối ưu mô hình.

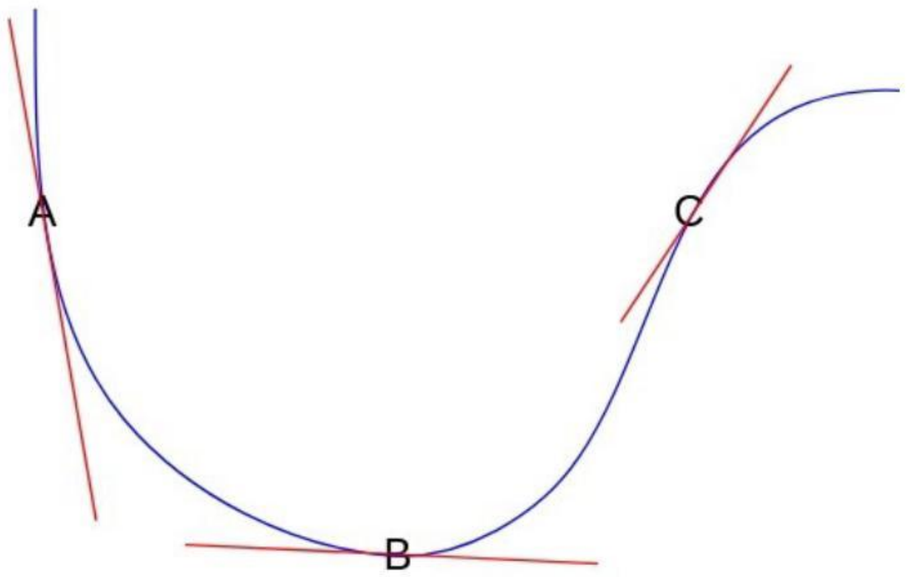


Đạo hàm

$$\frac{d}{dx}\left[(f(x))^n\right] = n(f(x))^{n-1} \cdot f'(x)$$

$$\frac{d}{dx}[f(g(x))] = f'(g(x))g'(x)$$

Interaction	Overall Change
Addition	$(f+g)' = f' + g'$
Multiplication	$(f \cdot g)' = f \cdot dg + g \cdot df$
Powers	$(x^n)' = \frac{d}{dx}x^n = nx^{n-1}$
Inverse	$\left(\frac{1}{x}\right)' = -\frac{1}{x^2}$
Division	$\left(\frac{f}{g}\right)' = \left(df \cdot \frac{1}{g}\right) + \left(\frac{-1}{g^2}dg \cdot f\right)$



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRUNG TÂM TIN HỌC

Backpropagation với Pytorch

❑ Ví dụ đơn giản về back propagation

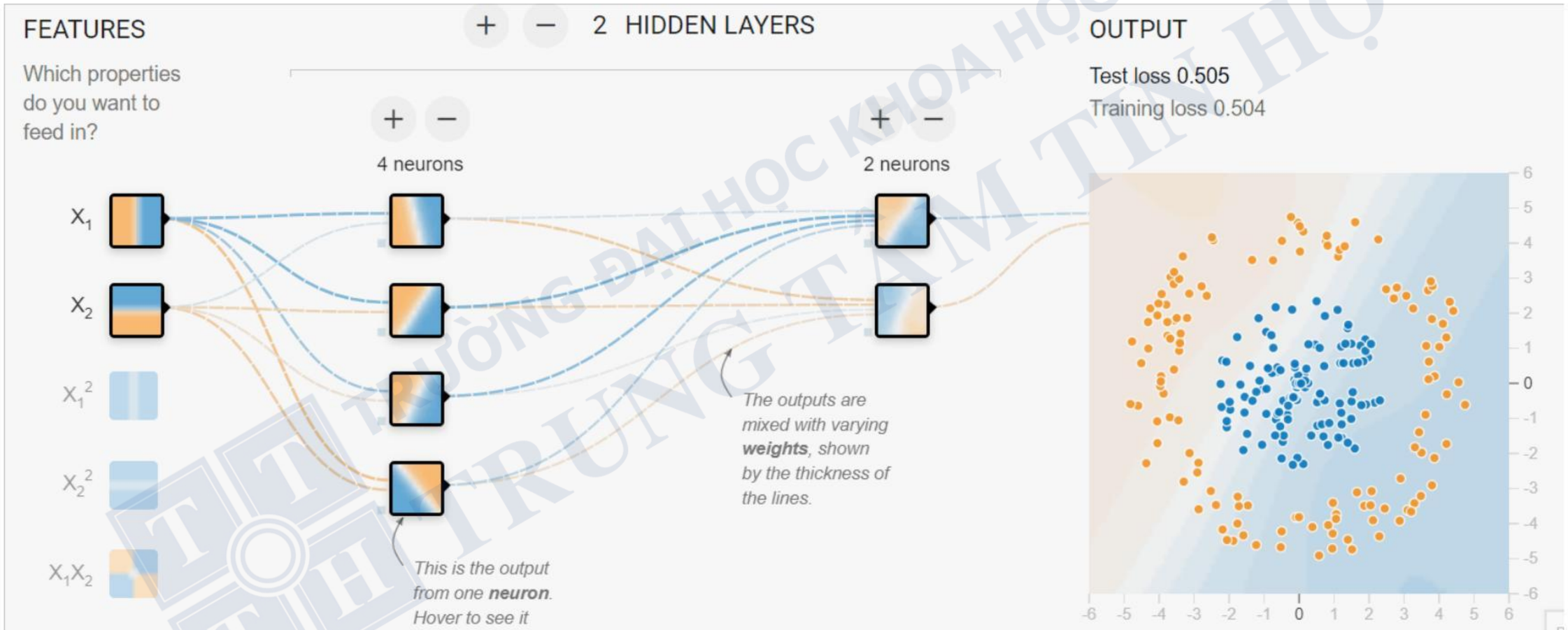
```
import torch

x = torch.tensor(-3., requires_grad=True)
y = torch.tensor(5., requires_grad=True)
 z = torch.tensor(-2., requires_grad=True)

q = x + y
f = q * z

f.backward()

print("Gradient of x is: " + str(x.grad))
print("Gradient of y is: " + str(y.grad))
print("Gradient of z is: " + str(z.grad))
```

```
Gradient of z is: tensor(2.)
Gradient of y is: tensor(-2.)
Gradient of x is: tensor(-2.)
```

Thực hiện **back propagation**

Neural Network & Non-linear Regression

Ví dụ về neural network với bài toán non-linear.



Code Demo

DEMO

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRUNG TÂM TIN HỌC

Q&A

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TRUNG TÂM TIN HỌC