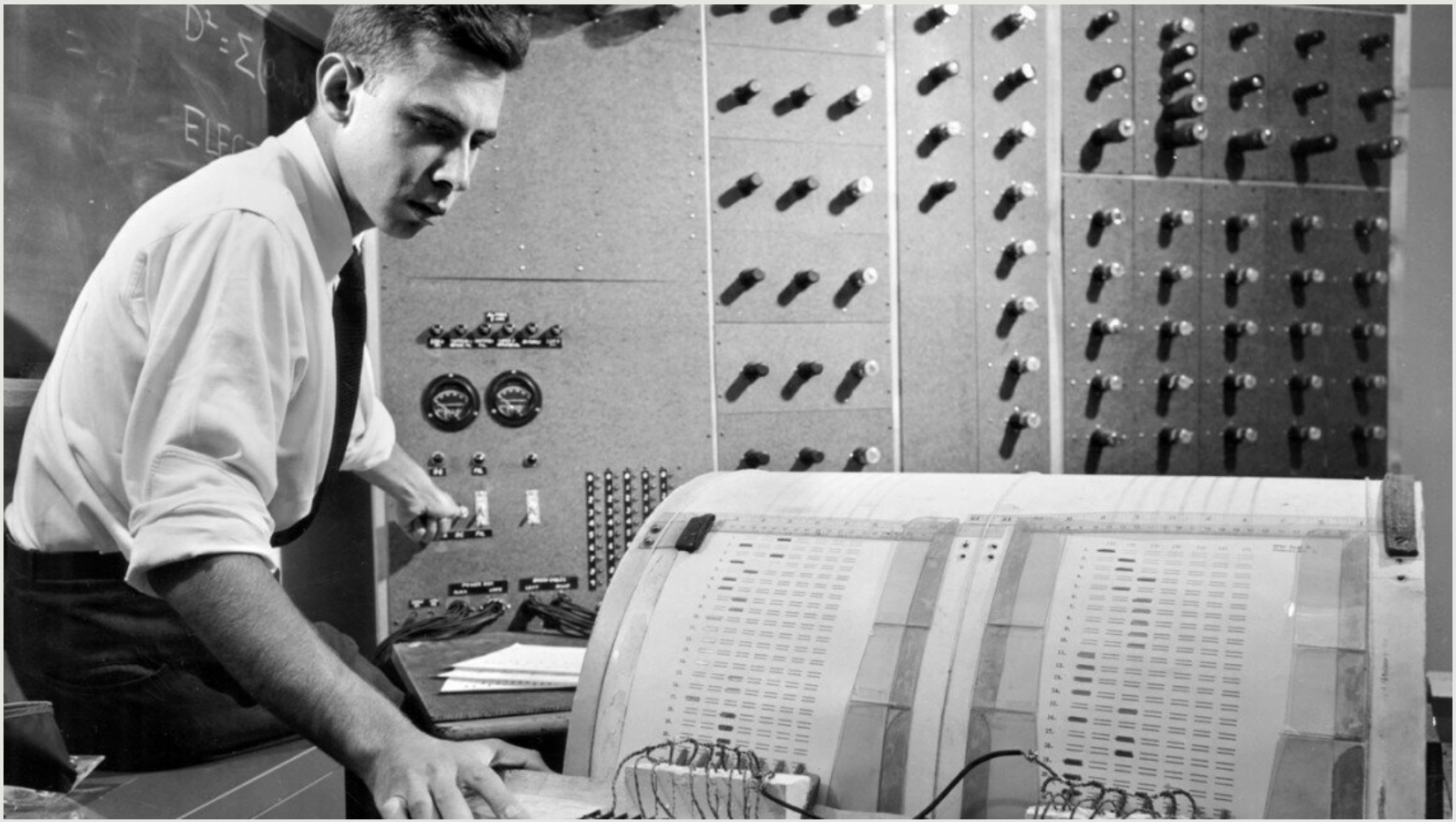
NEURAL NETWORKS

An Introduction to Neural Networks and Deep Learning

INTRODUCTION

The main idea and research originated in the late 1950s (F. Rosenblatt) with the “Mark I Perceptron”: the goal was to automatically detect capital letters.



Frank Rosenblatt ’50, Ph.D. ’56, works on the “electronic profile analyzing computer” – a precursor to the perceptron.

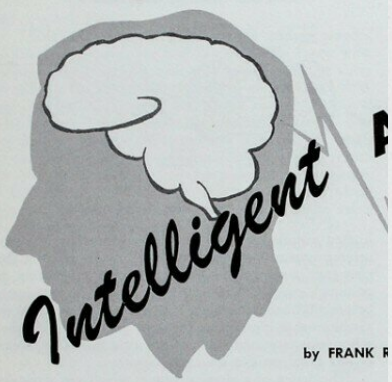
INTRODUCTION

Vol. VI, No. 2, Summer 1958

research trends

CORNELL AERONAUTICAL LABORATORY, INC., BUFFALO 21, NEW YORK

The Design of an Intelligent



by FRANK R

Introducing the perceptron - recognizes, remembers, and r

STORIES about the creation of machines having human qualities have long been a fascinating province in the realm of science fiction. Yet we are now about to witness the birth of such a machine — a machine capable of perceiving, recognizing, and identifying its surroundings without any human training or control.

Development of that machine has stemmed from a search for an understanding of the physical mechanisms which underlie human experience and intelligence. The question of the nature of these processes is at least as ancient as any other question in western science and philosophy, and, indeed, ranks as one of the greatest scientific challenges of our time.

Our understanding of this problem has gone perhaps as far as had the development of physics before Newton. We have some excellent descriptions of the phenomena to be explained, a number of interesting hypotheses, and a little detailed knowledge about events in the nervous system. But we lack agreement on any integrated set of principles by which the functioning of the nervous system can be understood.

We believe now that this ancient problem is about to yield to our theoretical investigation for three reasons:

nizing Automaton), an internal research program which had been in progress for over a year at Cornell Aeronautical Laboratory, received the support of the Office of Naval Research. The program had been concerned primarily with the application of probability theory to

FIG. 1 — Organization of a biological brain. (Red areas indicate active cells, responding to the letter X.)

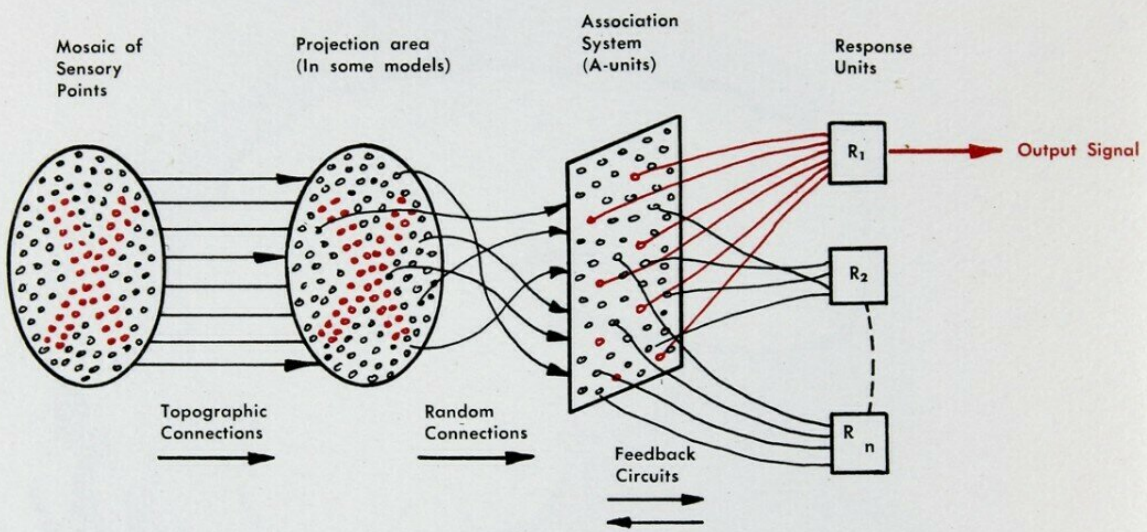


FIG. 2 — Organization of a perceptron.

INTRODUCTION

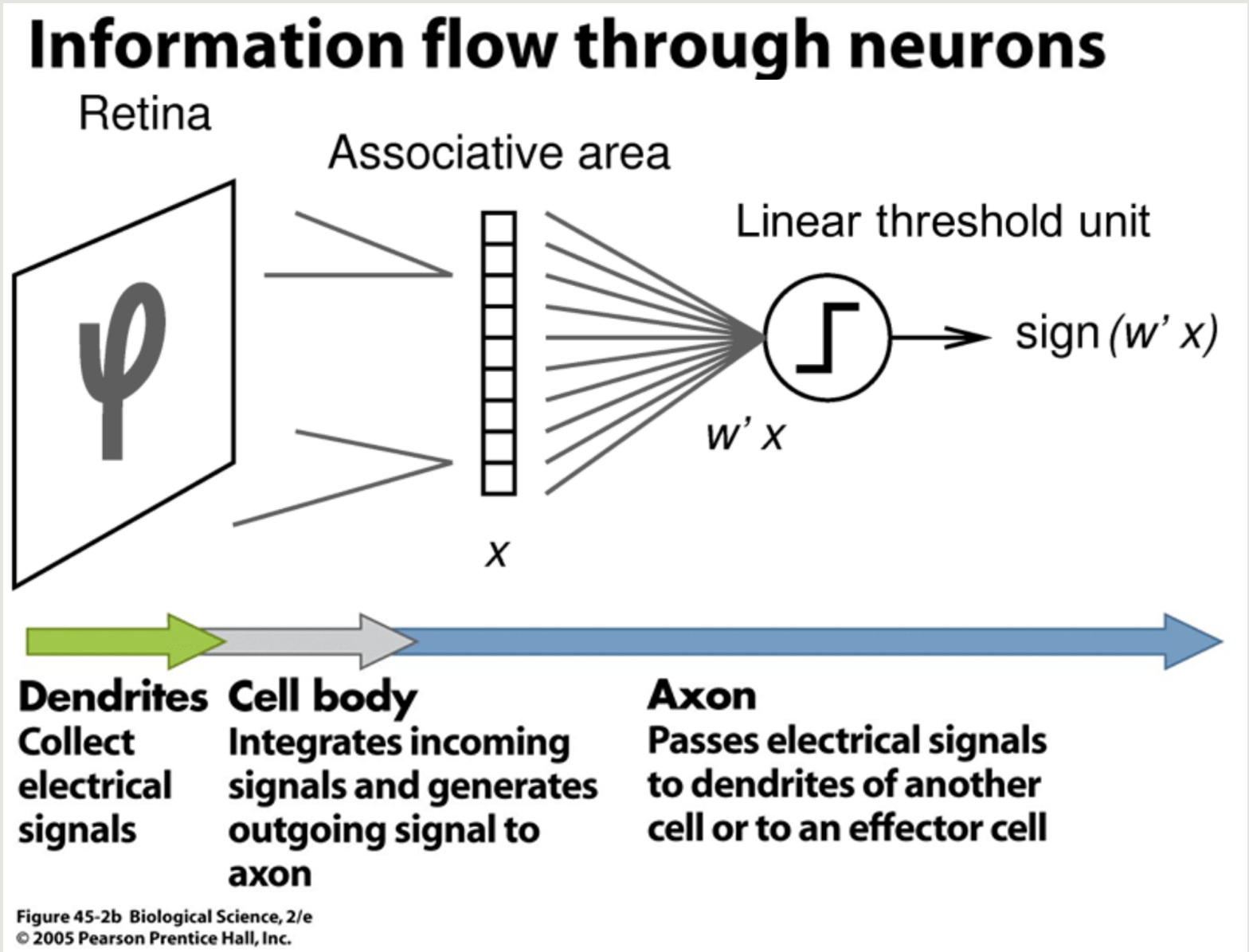
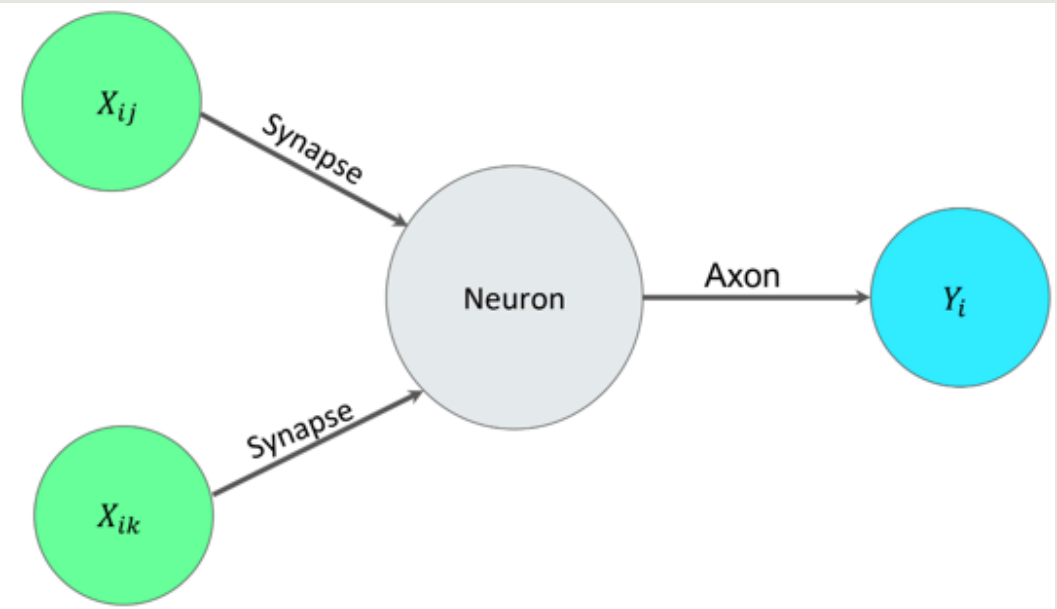
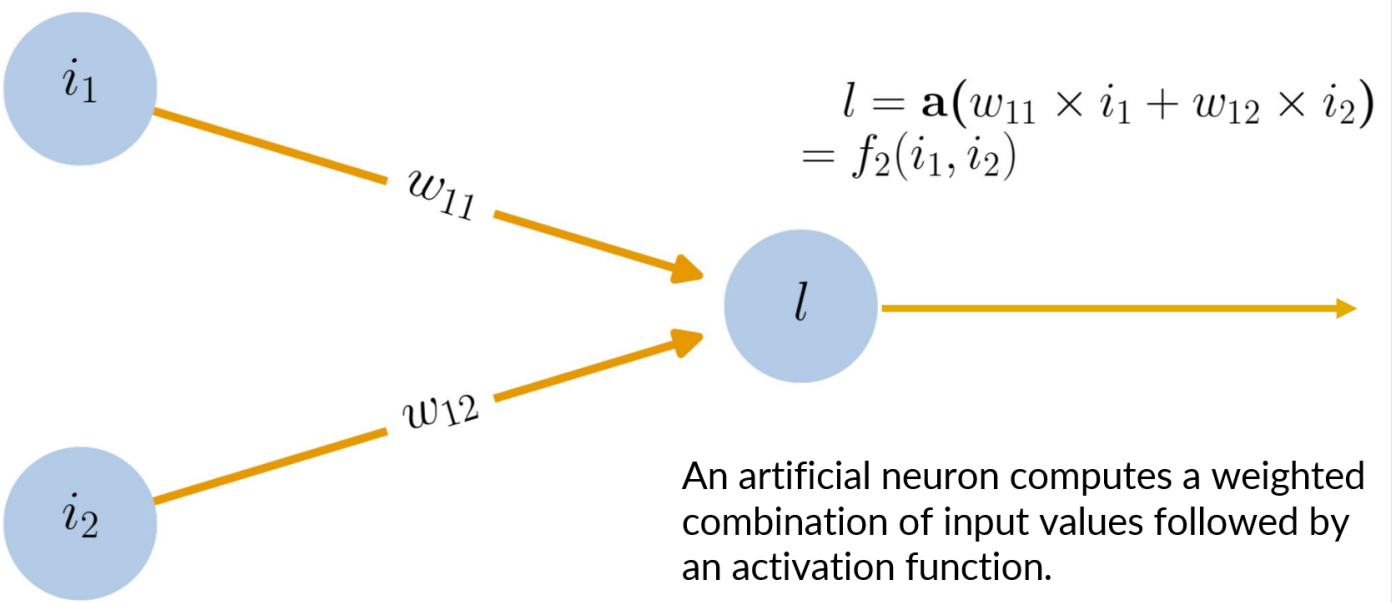


Figure 45-2b Biological Science, 2/e

THE ARTIFICIAL NEURON





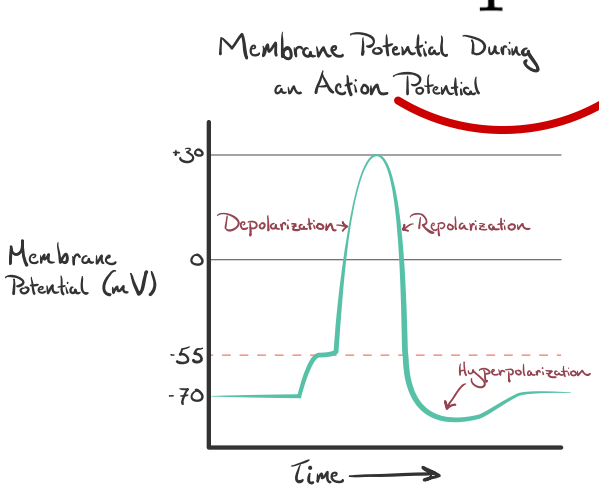
An artificial neuron computes a weighted combination of input values followed by an activation function.

ARTIFICIAL NEURON

What is an artificial neuron doing?

The artificial neuron is an activation function, e.g. sigmoid looking function. Consider the function of one neuron:

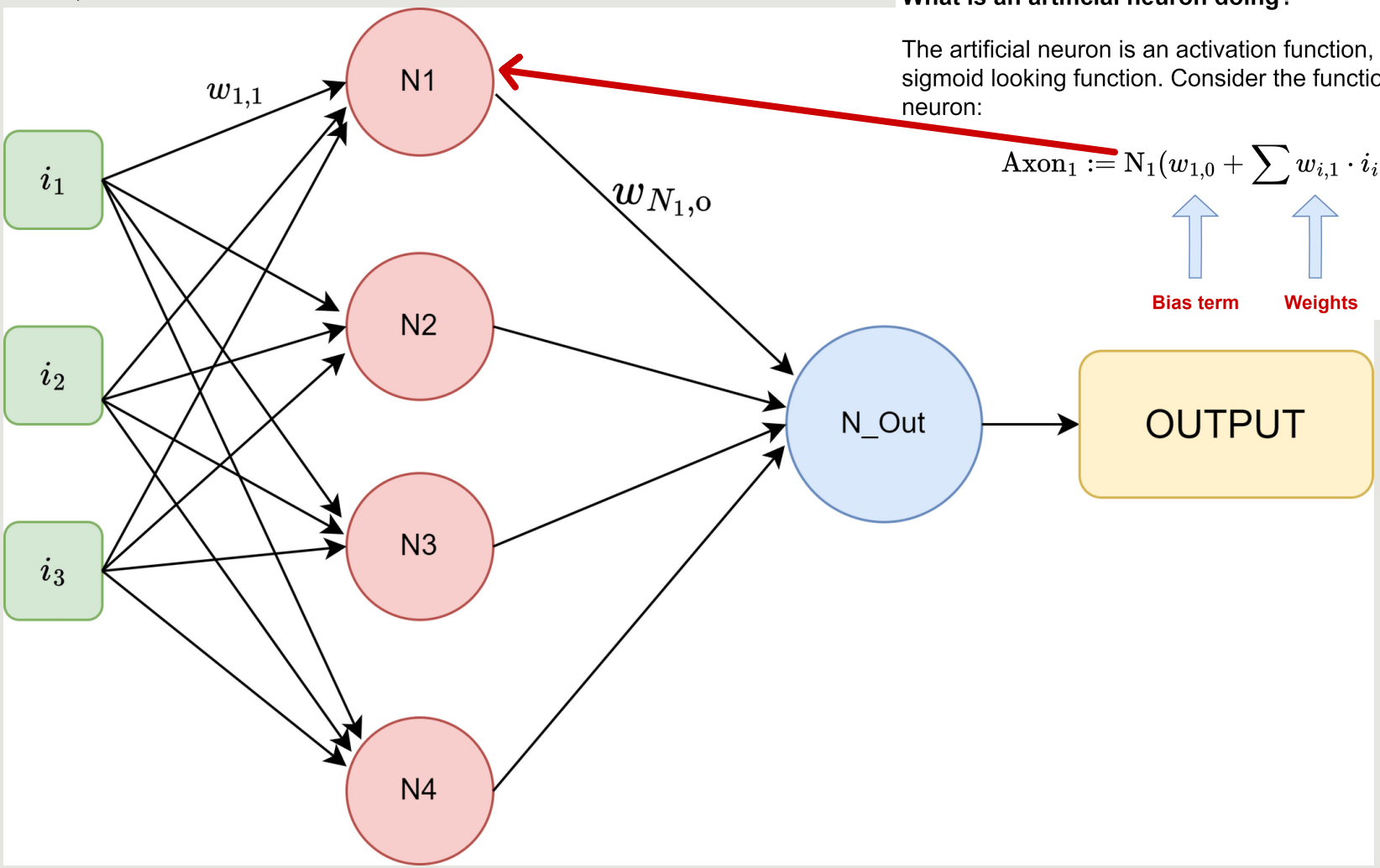
$$\text{Axon}_1 := \text{N}_1(w_{1,0} + \sum w_{i,1} \cdot i_i)$$



Bias term **Weights**

EXAMPLE OF NEURAL TOPOLOGY

In this example, we have one input layer with four neurons and one output neuron.



What is an artificial neuron doing?

The artificial neuron is an activation function, e.g. sigmoid looking function. Consider the function of one neuron:

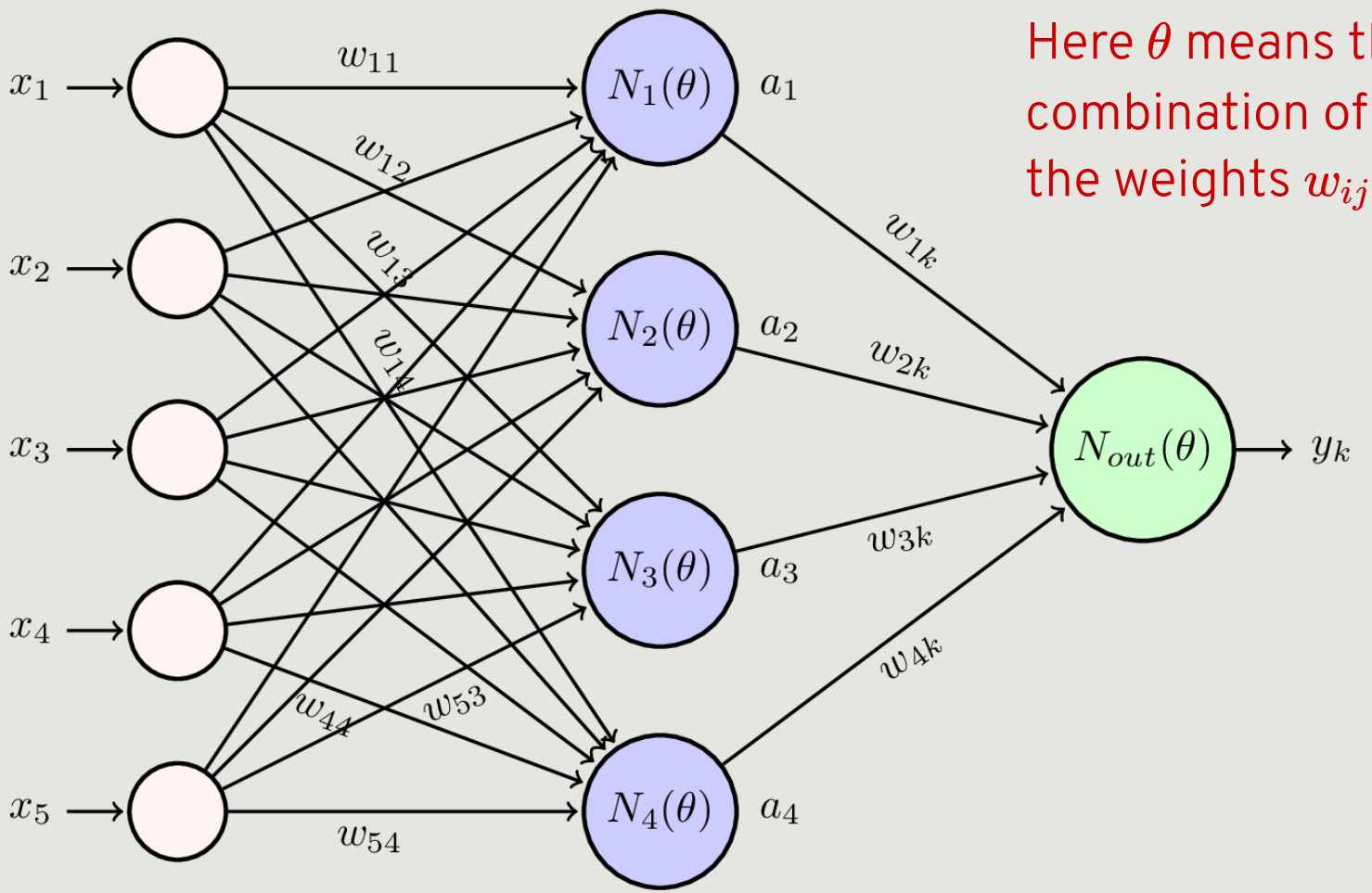
$$\text{Axon}_1 := \text{N}_1(w_{1,0} + \sum w_{i,1} \cdot i_i)$$

Bias term Weights

AN EXAMPLE OF THE STRUCTURE OF NEURAL NETS

Data x_i | $a_j = N_i(\sum_{i=1}^{N} w_{ij}x_i)$ | $y_k = N_{out}(\sum_{j=1}^{M} w_{jk}a_j)$

Input Layer | **Hidden Layer** | **Output Layer**

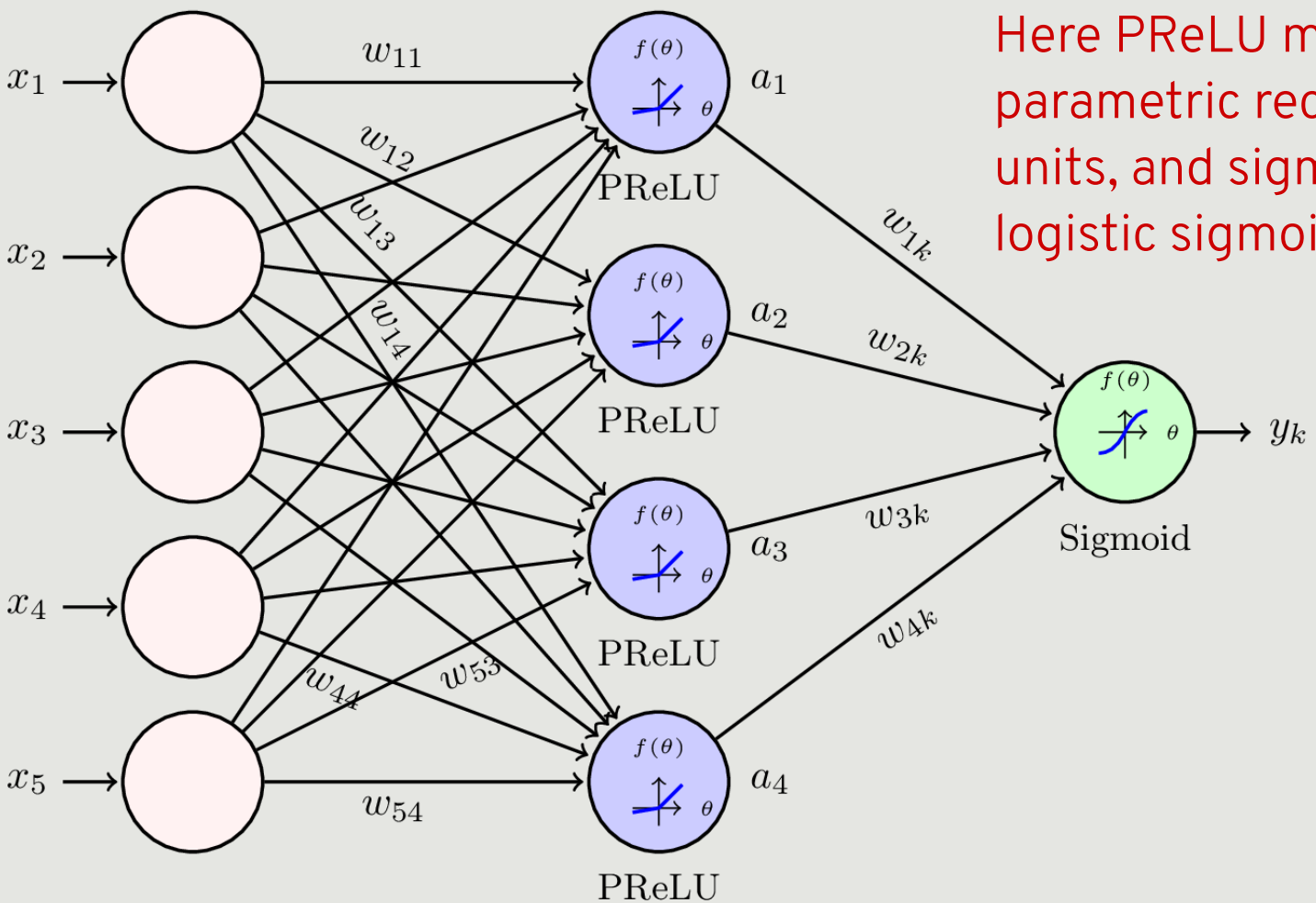


Here θ means the linear combination of the inputs by the weights w_{ij}.

AN EXAMPLE OF THE STRUCTURE OF NEURAL NETS

Data x_i | $a_j = f(\sum_{i=1}^{N} w_{ij} x_i)$ | $y_k = g(\sum_{j=1}^{M} w_{jk} a_j)$

Input Layer | **Hidden Layer** | **Output Layer**



Here PReLU means parametric rectified linear units, and sigmoid is the logistic sigmoid.

AN EXAMPLE OF NEURAL NETWORK IN PYTORCH

```
# Define a simple neural network
class NeuralNet(nn.Module):
    def __init__(self, n_features):
        super(NeuralNet, self).__init__()
        self.fc1 = nn.Linear(n_features, 5).double()
        self.prelu = nn.PReLU().double()
        self.fc2 = nn.Linear(5, 3).double()

    def forward(self, x):
        x = self.prelu(self.fc1(x))  # Apply PReLU activation
        x = self.fc2(x)
        return x

# Initialize the model, loss function, and optimizer
model = NeuralNet(X_train.shape[1])
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)
```

AN EXAMPLE OF NEURAL NETWORK IN PYTORCH

```
# Train the model
num_epochs = 100
for epoch in range(num_epochs):
    # a peculiar aspect of Pytorch -> you put the model in a "training" st
    model.train()
    for X_batch, y_batch in train_loader:
        # this resets the optimizer before each calculation of the directi
        optimizer.zero_grad()
        # do a forward propagation
        outputs = model(X_batch)
        # use the criterion to compute the loss of the batch
        loss = criterion(outputs, y_batch)
        # here we backpropagate to update the weigths
        loss.backward()
        optimizer.step()

    if (epoch+1) % 10 == 0:
        print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')
```

NATURE-INSPIRED DESIGN

Species	# of Neurons
ant	250 K
bee	960 K
frog	16 M
rat	200 M
cat	760 M
owl	1 B
dog	2.8 B
lion	4.7 B
monkey	30 B
human	86 B

Check out the playground.

WHAT IS THE MACHINE LEARNING?

Assuming a choice of activation functions and network topology, the machine must learn the *optimal* weights.

Critical Thinking: How can the evil machine do that?

Backpropagation: Gradient-based optimization of the weights.

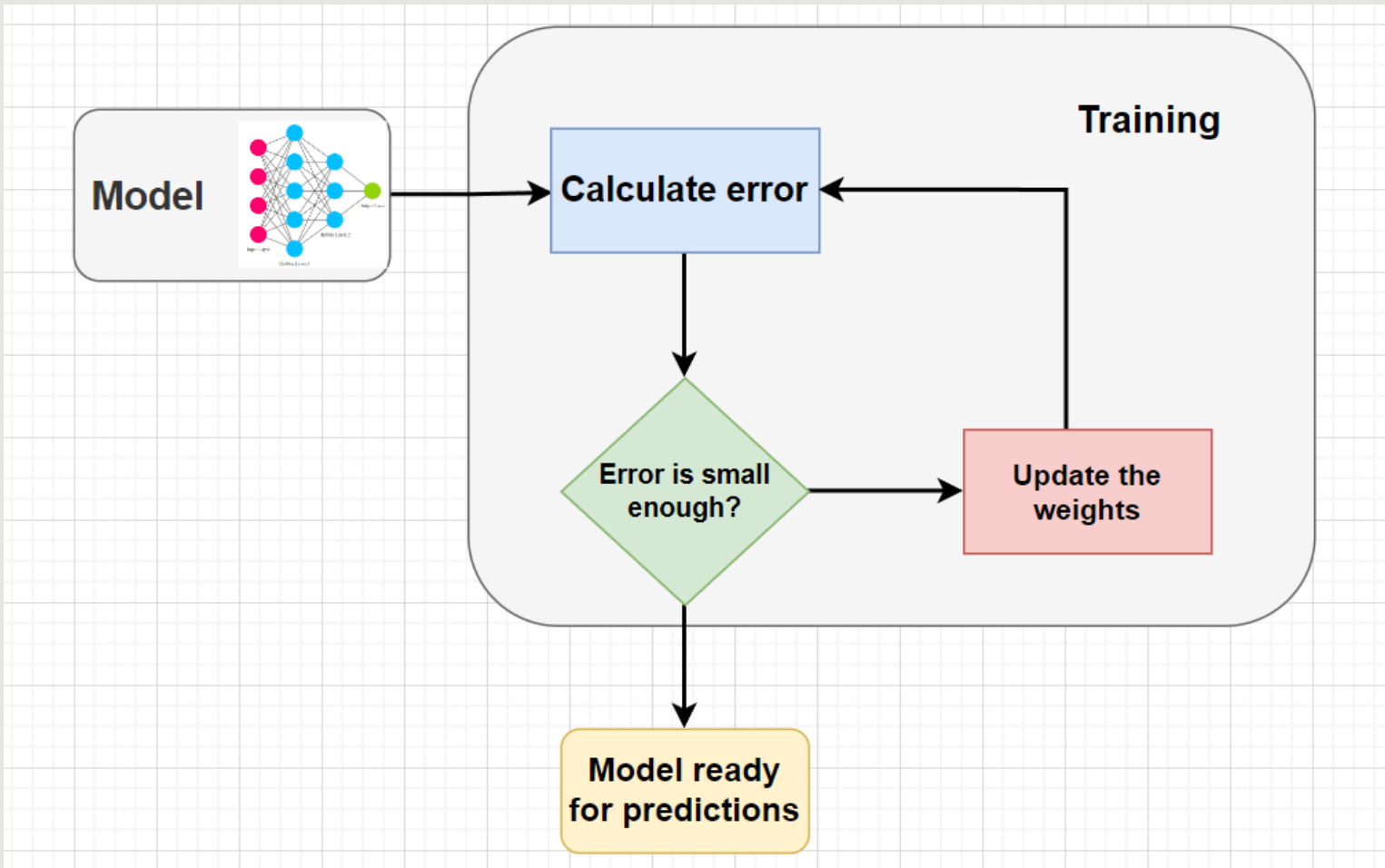
Meet Mr. Robot!

- D. Rumelhart, G. Hinton and R. Williams were able to achieve the first back-propagation network (Seminal paper "Learning Representations by back-propagating errors" Nature Vol. 323, 1986).
- 1990, P. Werbos, "Backpropagation Through Time: What It Does and How to Do It"
- G. Hinton and R. Salakhutdinov examined the capability of neural nets for image recognition in 2006. Explosive research into neural nets from 2006 - today.

Reference: https://torontolife.com/life/ai-superstars-google-facebook-apple-studied-guy/

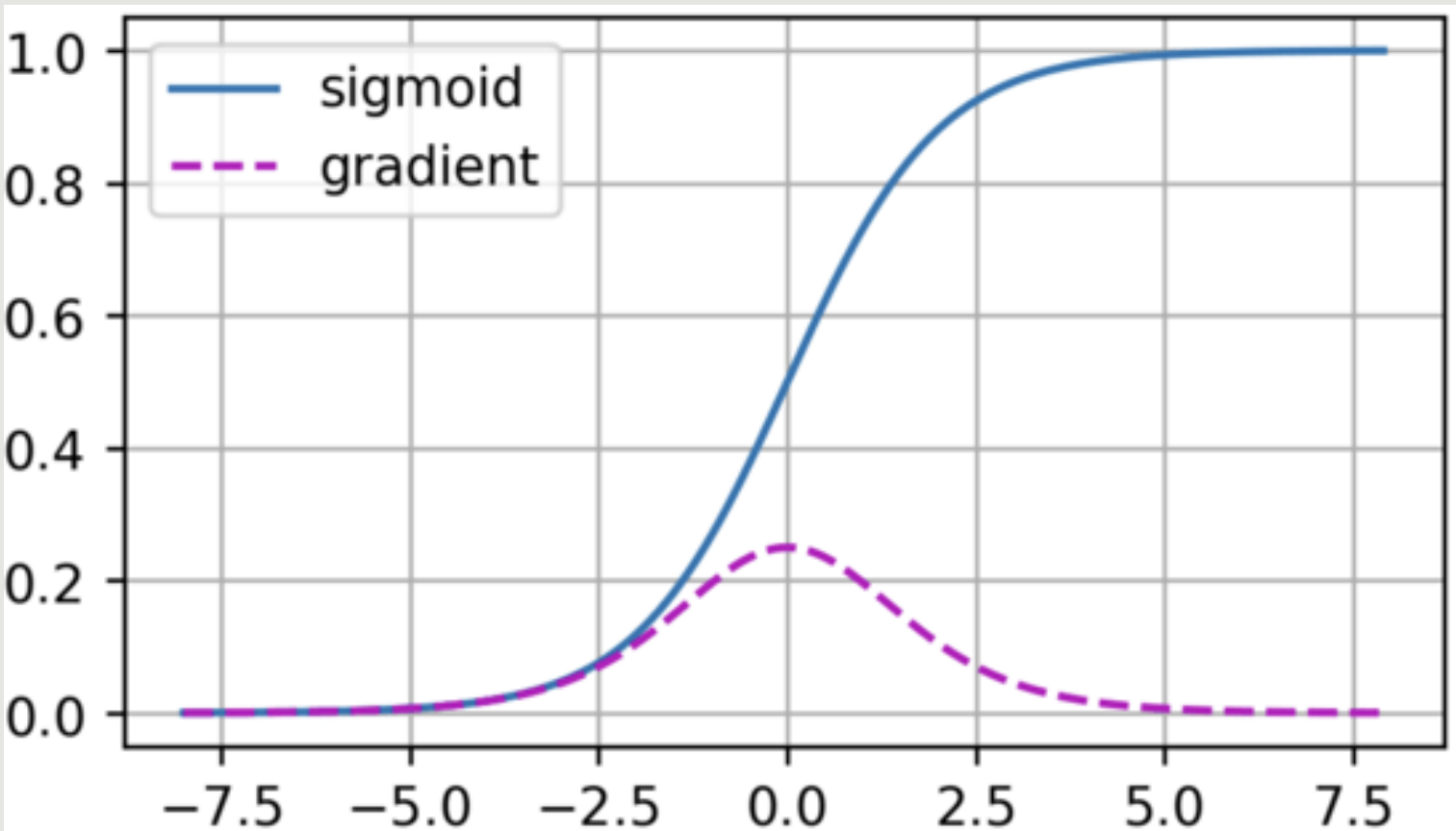
BACKPROPAGATION

Backpropagation is the process of updating the weights using an algorithm based on the gradient of the loss function. Examples include Root Mean Square Propagation and Adaptive Momentum Estimation (ADAM).



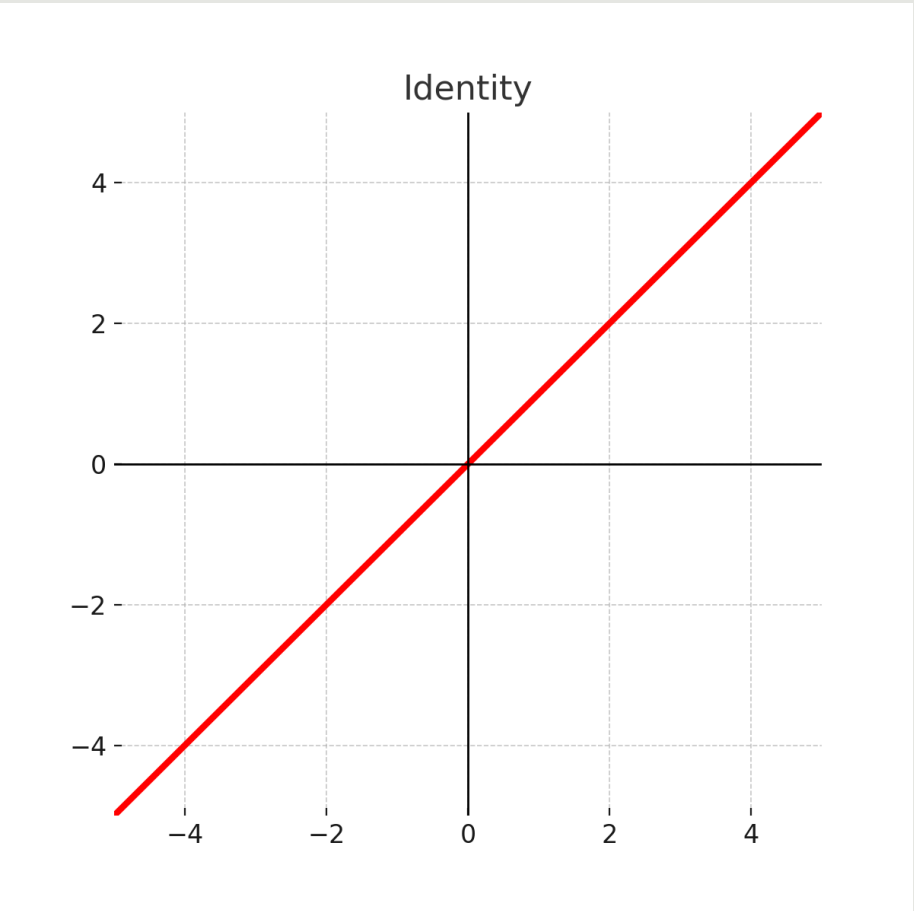
PROBLEMS WITH BACKPROPAGATION

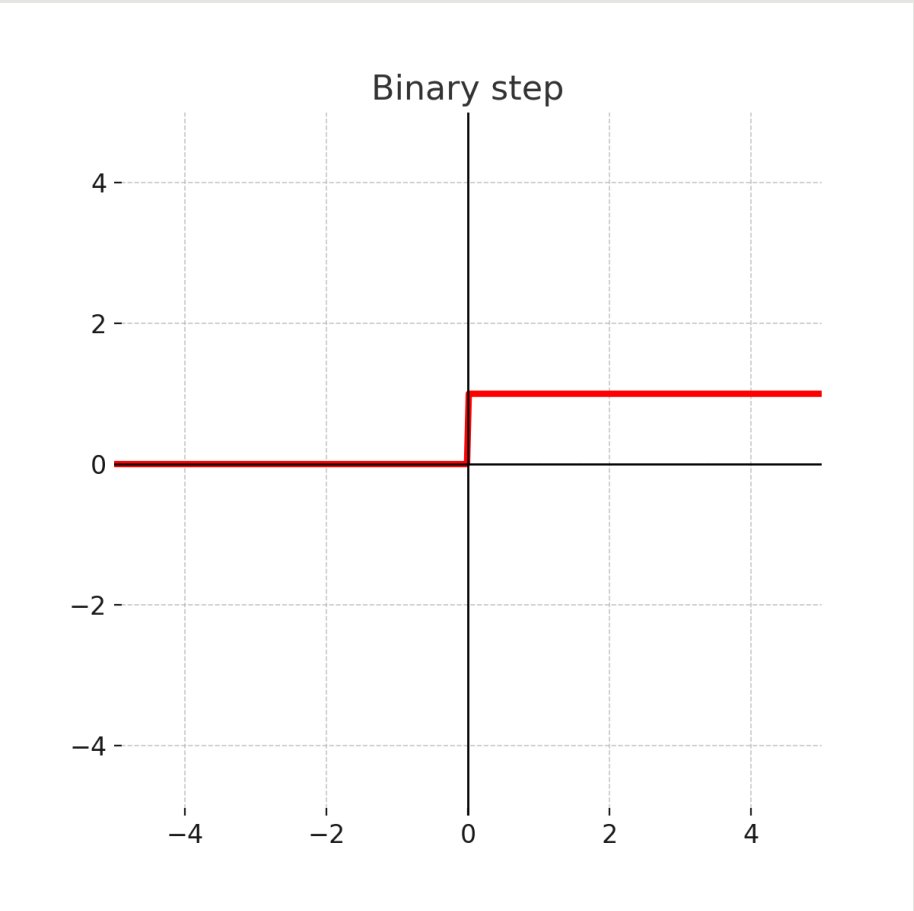
The *vanishing gradient* problem:



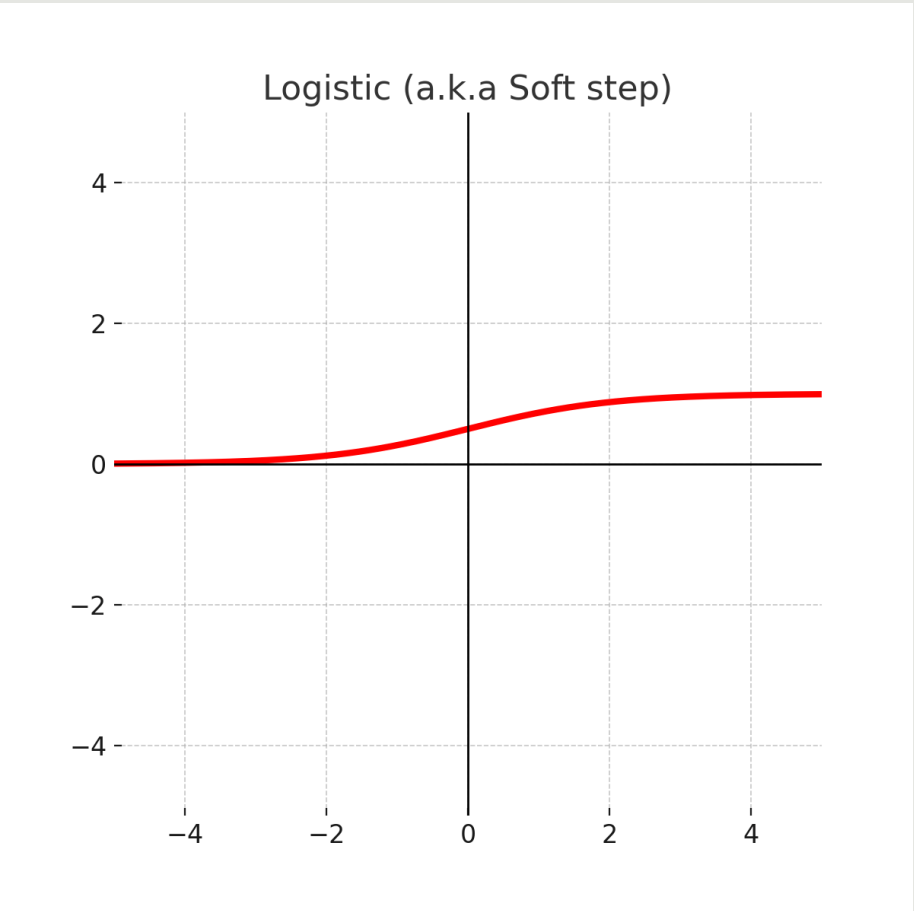
The problems can be remedied when experimenting with particular choices of activation functions.
https://pytorch.org/docs/main/nn.html#non-linear-activations-weighted-sum-nonlinearity

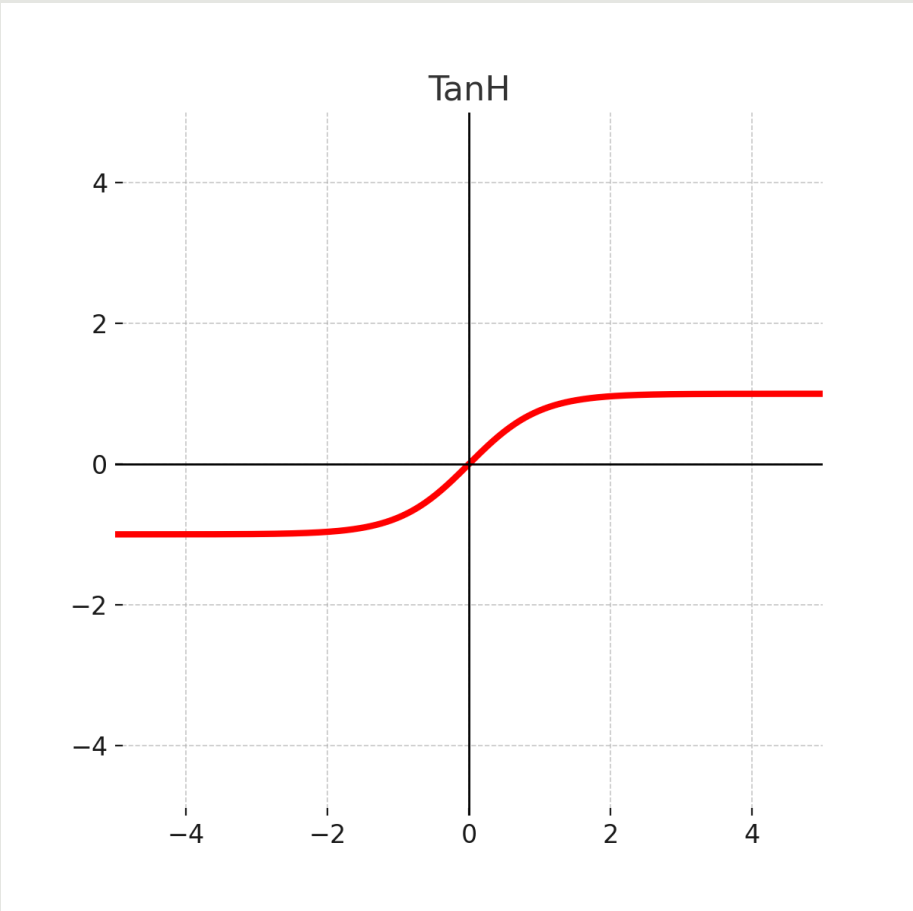
ACTIVATION FUNCTIONS -VISUAL PERSPECTIVE



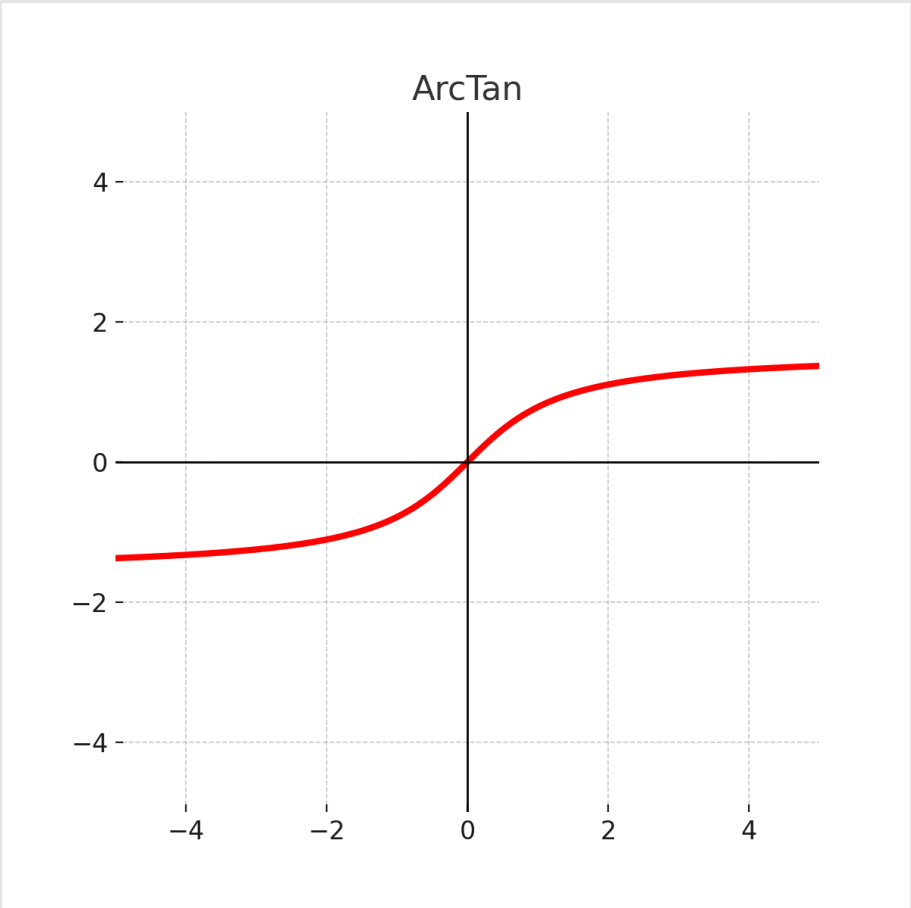


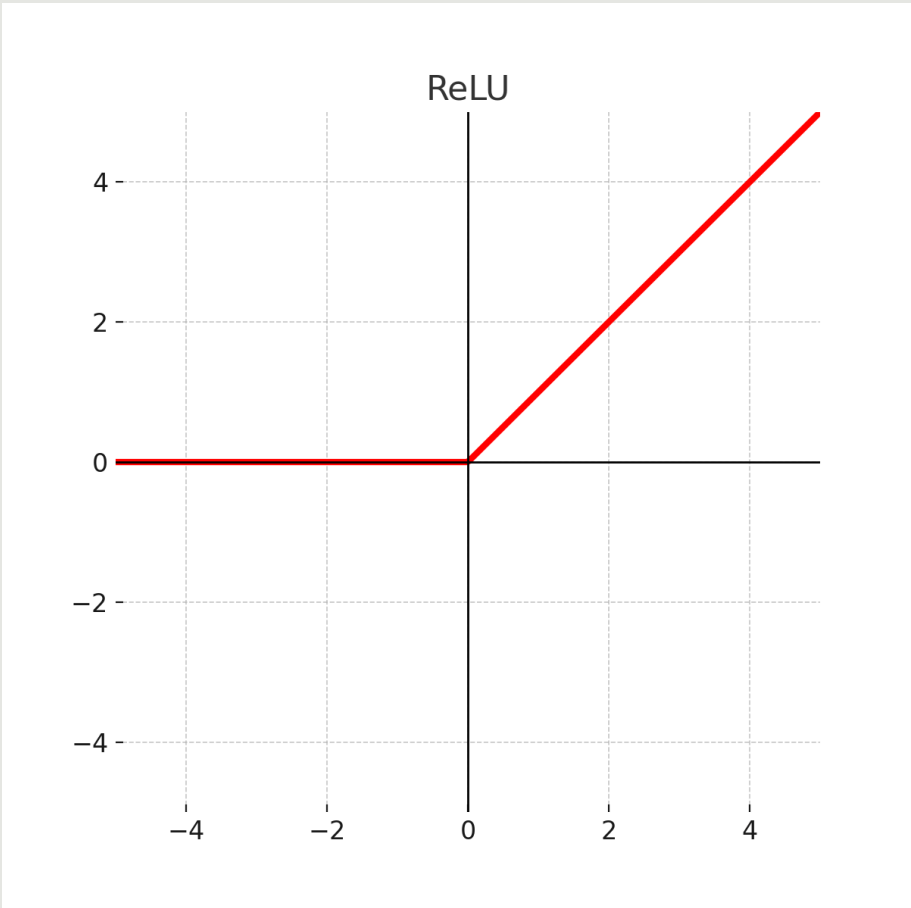
ACTIVATION FUNCTIONS -VISUAL PERSPECTIVE



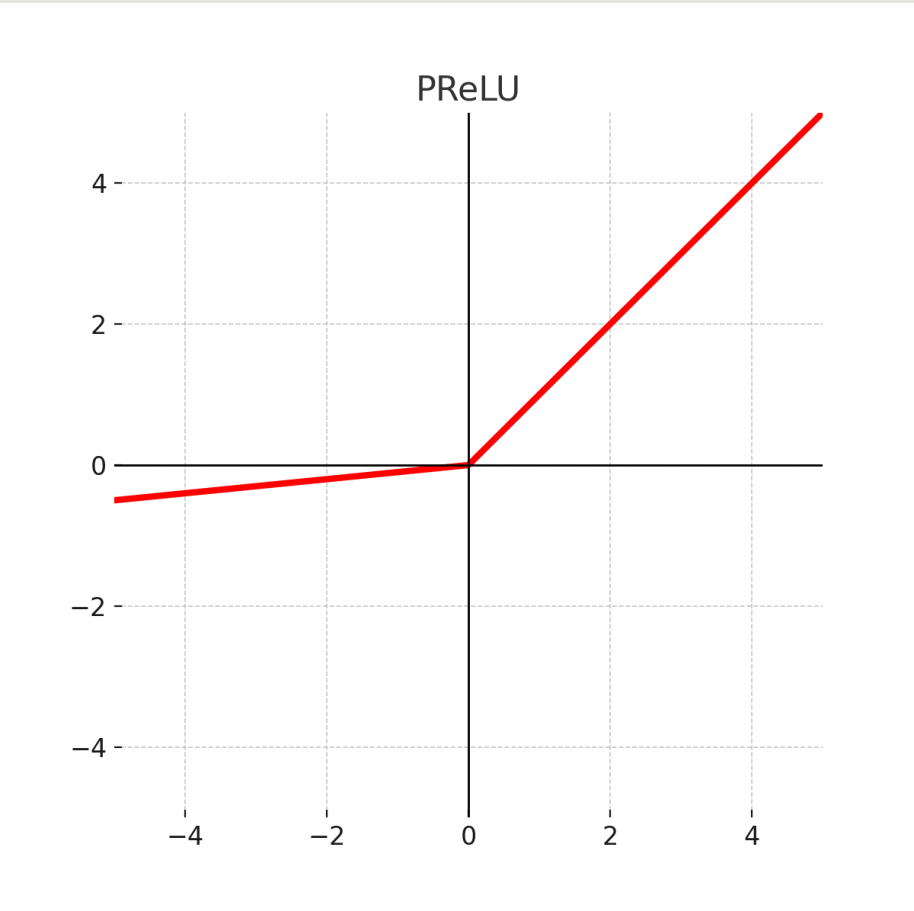


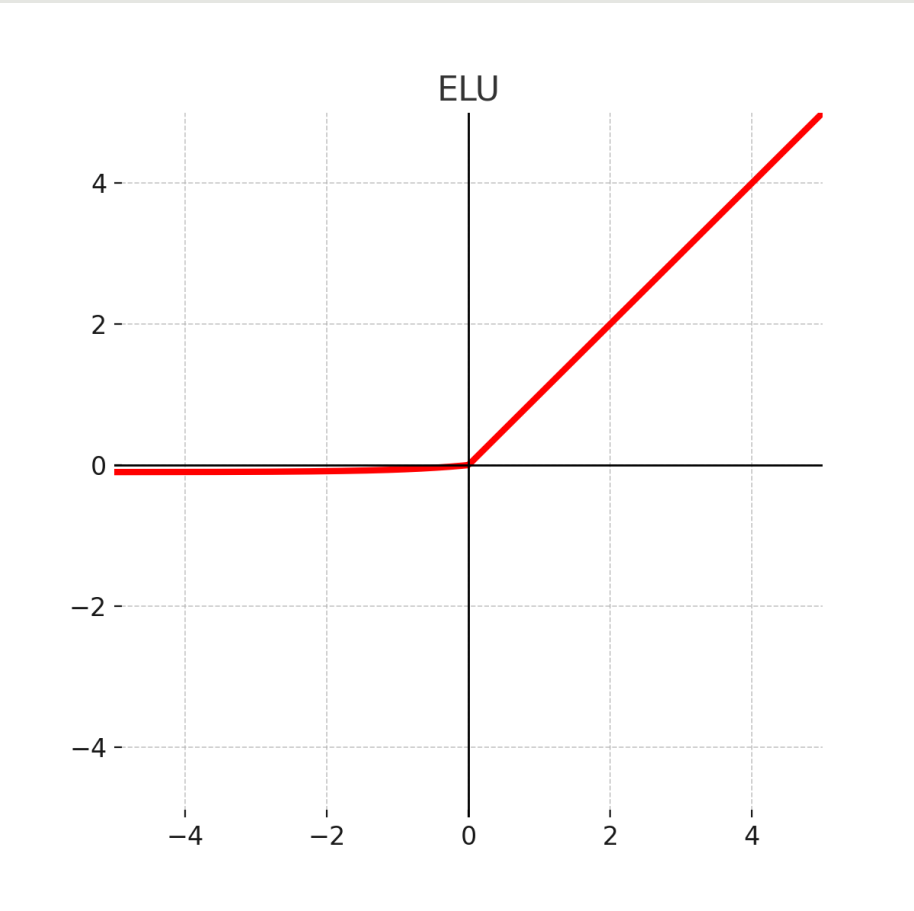
ACTIVATION FUNCTIONS -VISUAL PERSPECTIVE



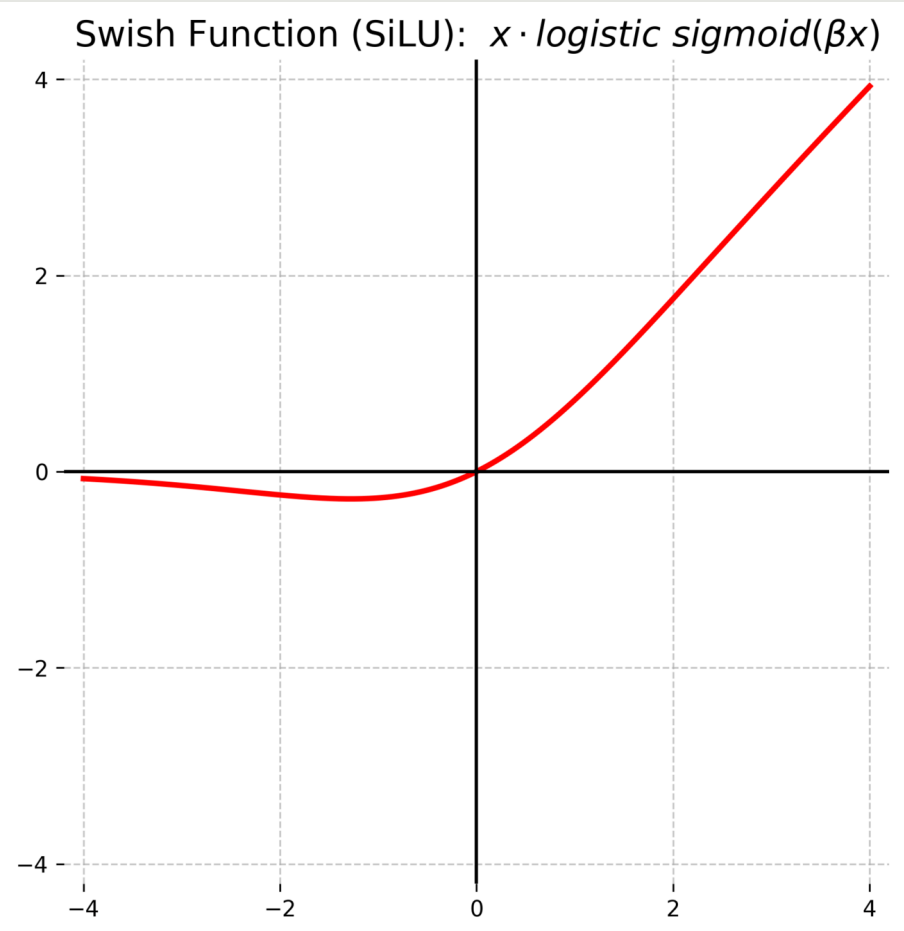


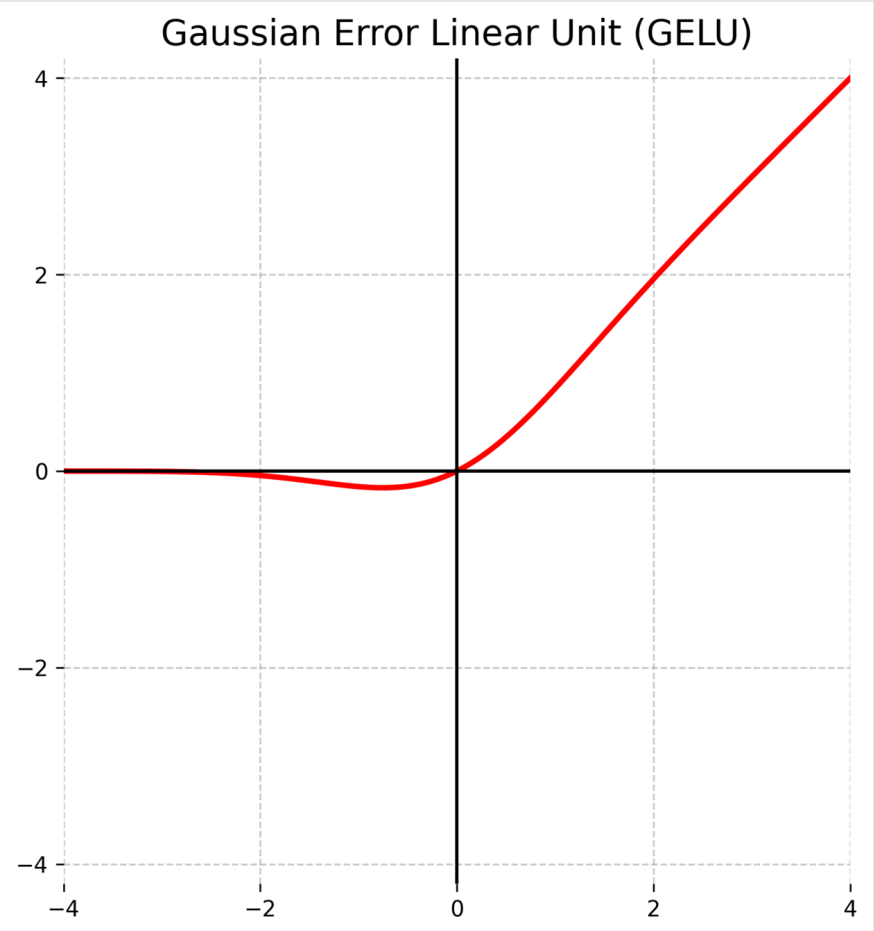
ACTIVATION FUNCTIONS -VISUAL PERSPECTIVE





ACTIVATION FUNCTIONS -VISUAL PERSPECTIVE





MORE LEARNING MATERIALS

The first 2 are required. The last 2 are optional but highly recommended, even if you have not had any calculus or linear algebra!

1. But what is a Neural Network?
2. Gradient descent, how neural networks learn
3. What is backpropagation really doing?
4. Backpropagation calculus

TEST YOUR UNDERSTANDING

1. What is a multilayer perceptron?
2. What is a hidden layer?
3. The network in the video was on the small-ish side, having only 2 hidden layers with 16 neurons each. How many total parameters (i.e. weights and biases) have to be determined during the training process for this network?
4. Do you understand the concept of gradient descent?