University of Lancashire

Lecture 12

Basics of Deep Learning

Course overview

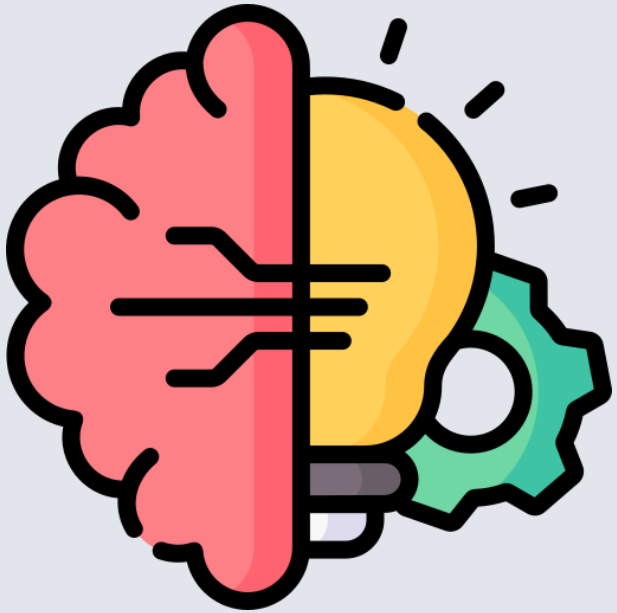
- **Learning Objectives**
  - Introduce and familiarise you with the Basics of Deep Learning
  - Learning how multi-perceptrons works
  - Understanding about the problems in each NN followed by CNN.

- **Important Directions**



Today's Contents

- Abstract Neuron Model
- Multilayer Perceptrons (MLPs)
- Feedforward Neural Networks
- Backpropagation Algorithm
- Importance of Weight Initialization and Learning Rate
- Problems with Fully Connected Deep Neural Networks
- Convolutional Neural Networks



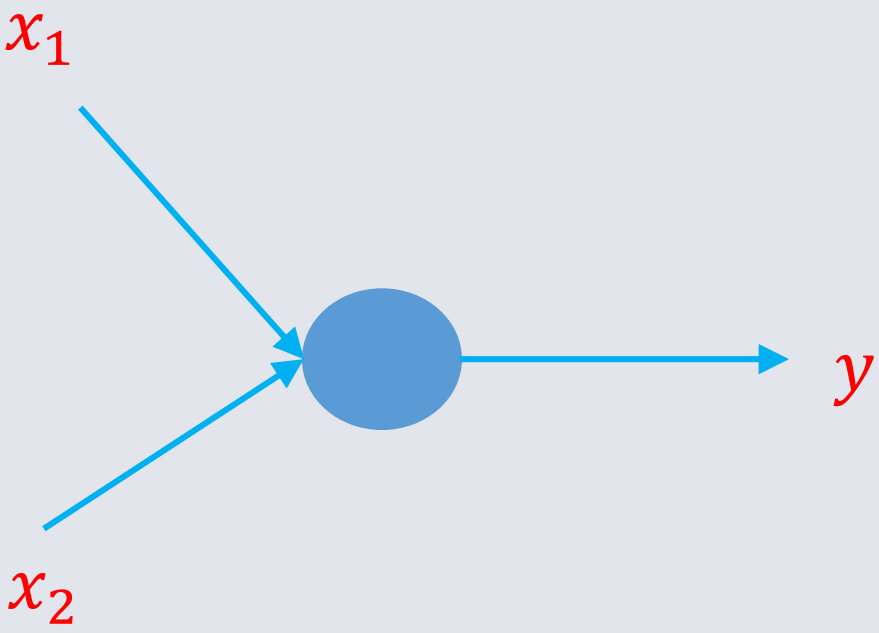
Abstract Neuron Model

Neuron in a model processes information by taking **inputs**,

applying weights to them, adding a **bias**, and then finding

whether it should “activate" or not based on the result.

$$y = \theta^T x$$



x is the vector of inputs to the neuron,

θ is the vector of weights (also called a pattern or filter),

$\theta^T x$ is the dot product between weights and inputs.

$$y = \theta^T x + b$$

A bias is added to the weighted sum. This bias helps the neuron adjust its threshold for activation.

Abstract Neuron Model

Neuron in a model processes information by taking **inputs**,

applying weights to them, adding a **bias**, and then finding

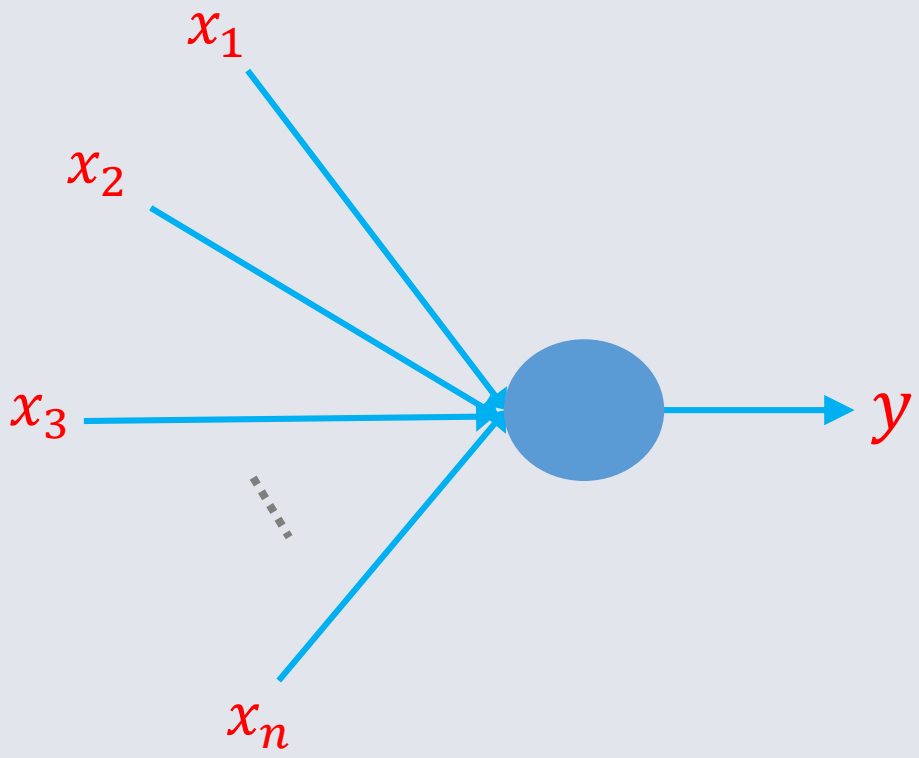
whether it should “activate" or not based on the result.

$$y = \theta^T x$$

x is the vector of inputs to the neuron,

θ is the vector of weights (also called a pattern or filter),

$\theta^T x$ is the dot product between weights and inputs.



$$y = \theta^T x + b$$

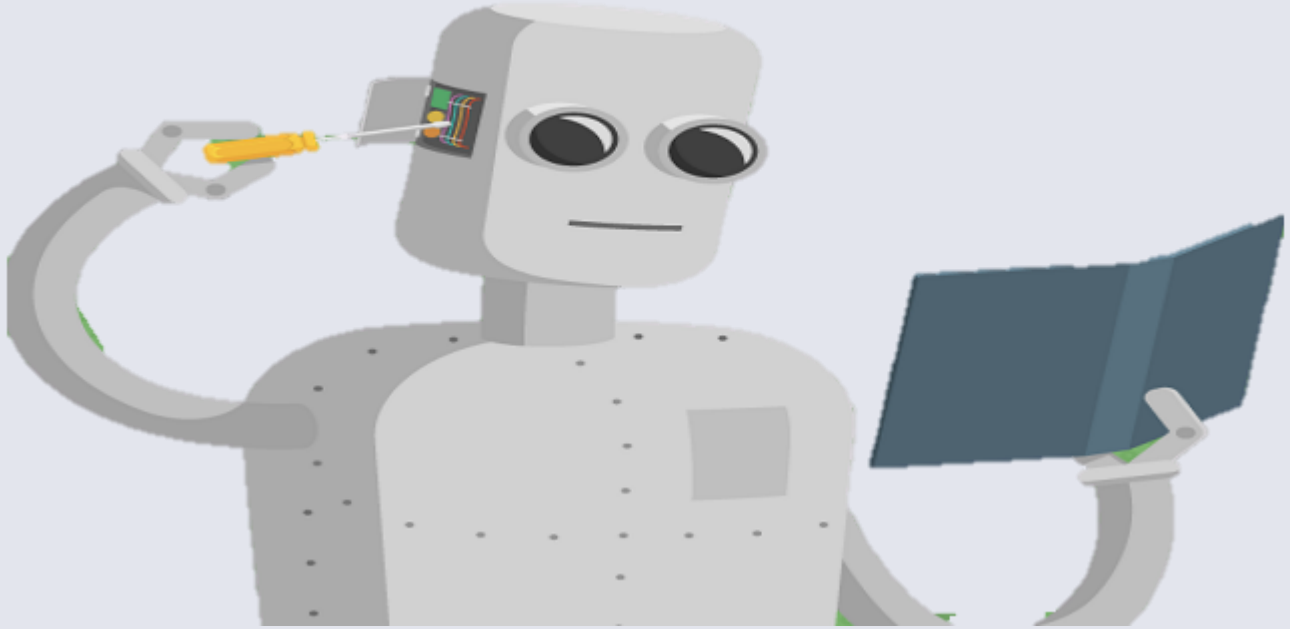
A bias is added to the weighted sum. This bias helps the neuron adjust its threshold for activation.

Artificial Neural Network

Problems with Single Perceptron?

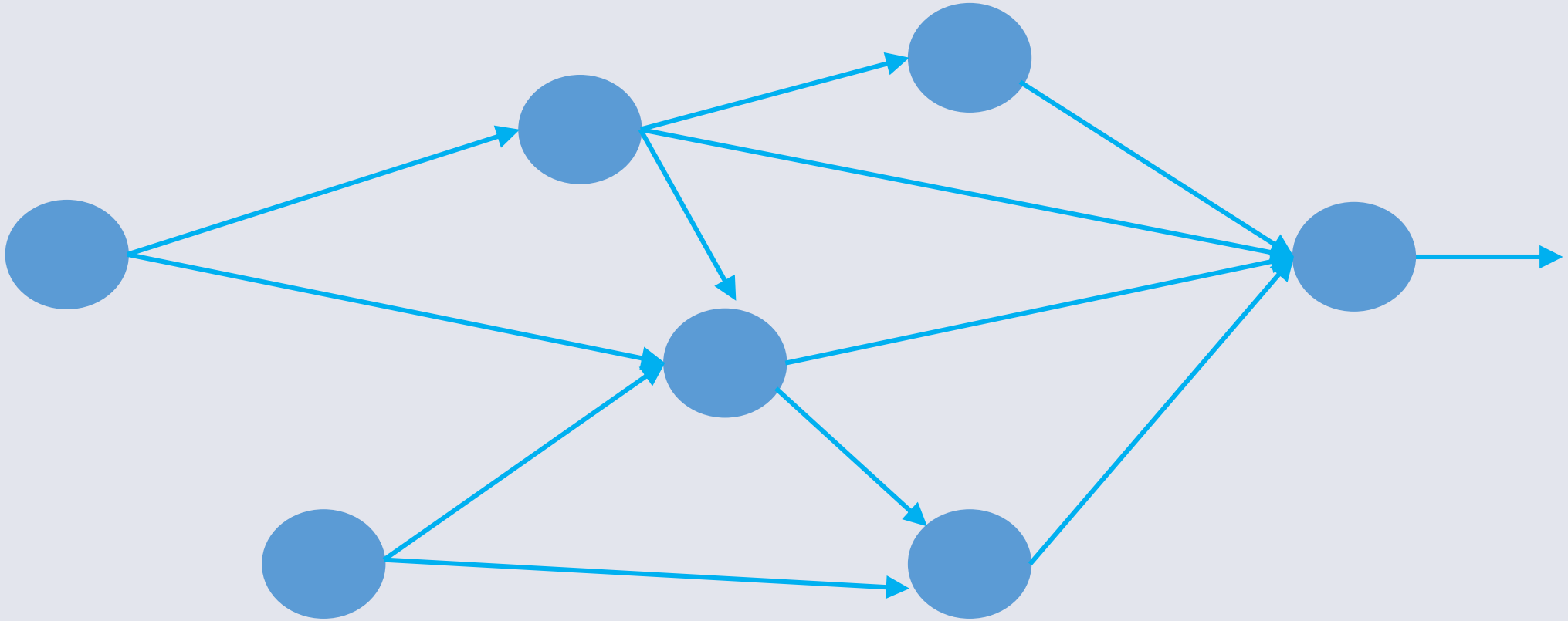
Single-layer perceptrons can only solve linearly separable problems. (Step function, Linear function)

MLPs with hidden layers can approximate complex functions using non-linear activation functions.



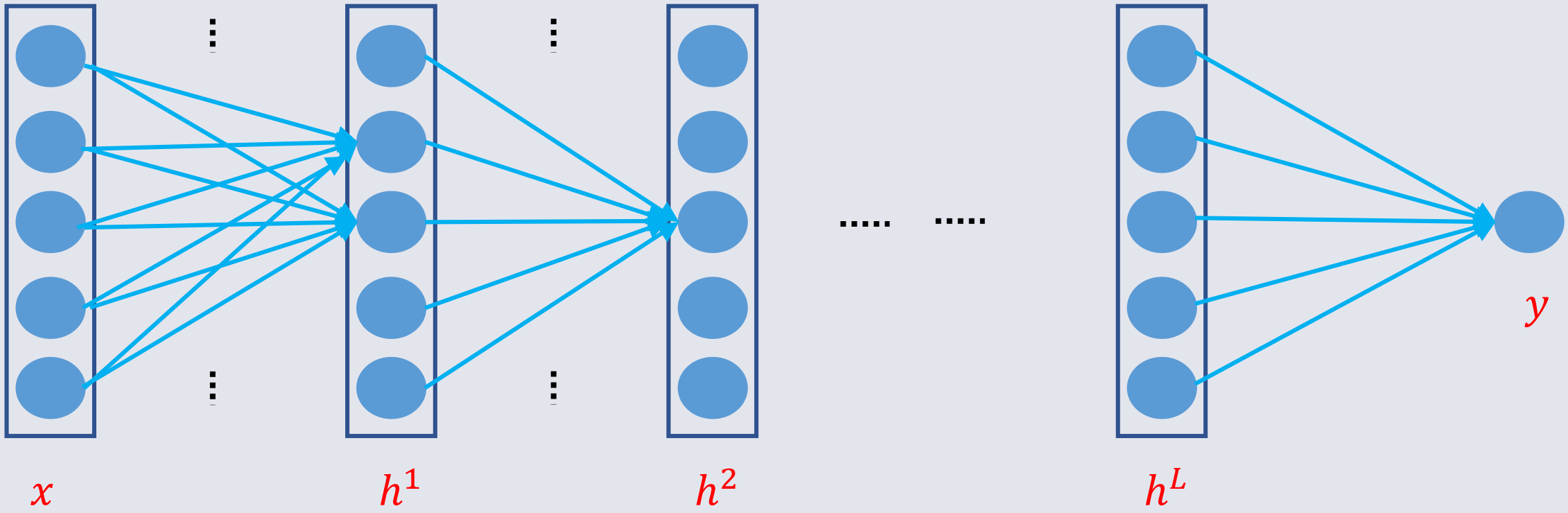
Artificial Neural Network

Combination of many neurons making MLP



Feedforward Propagation

Where data flows in one direction, from the input layer, through one or more hidden layers, to the output layer.



Feedforward Propagation

Where data flows in one direction, from the input layer, through one or more hidden layers, to the output layer.

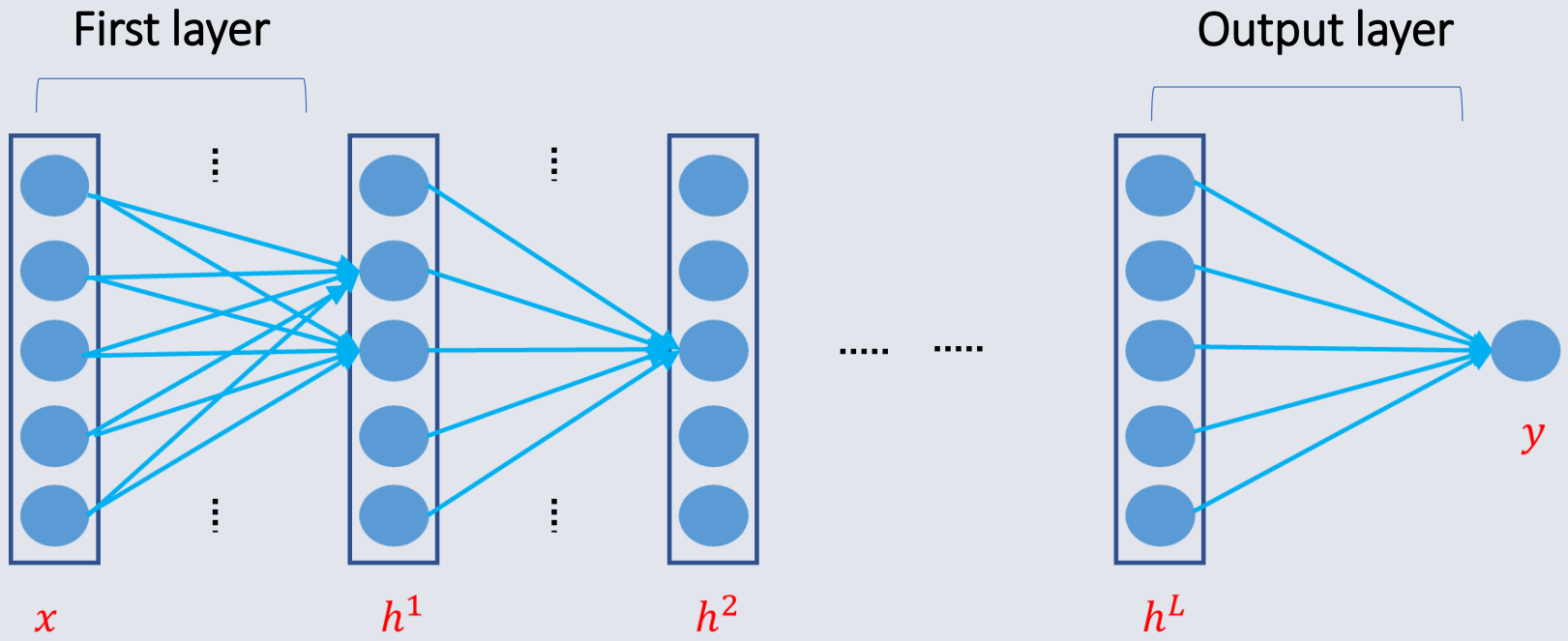
Components

Representations:

- Input
- Hidden variables

Layers/weights:

- Hidden layers
- Output layer



Feedforward Deep Network: Components

Input Layer

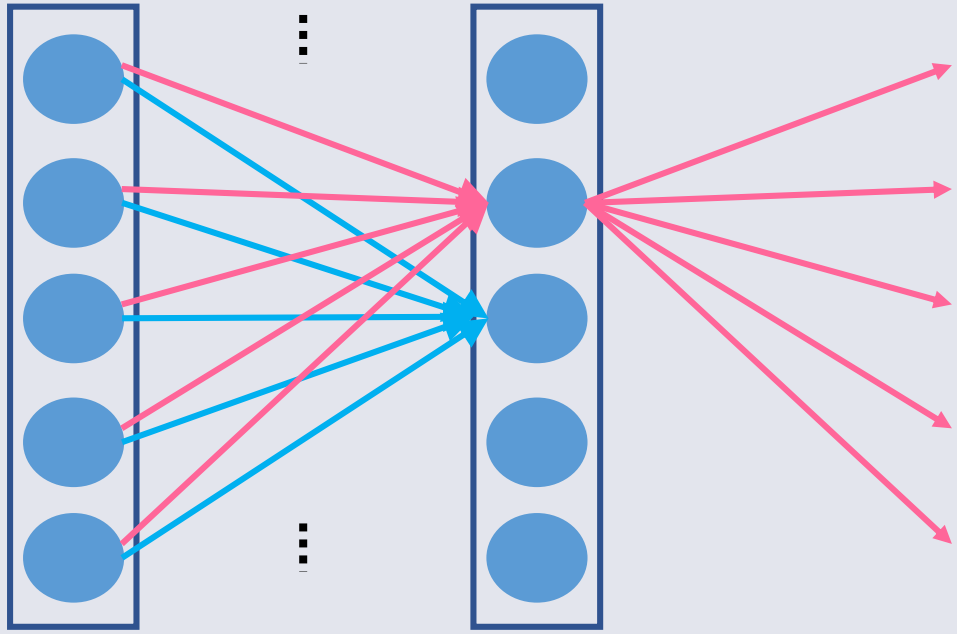
- The model receives **raw features** from the image.
- Represented as a vector

Expand

Feedforward Deep Network: Components

Hidden Layer

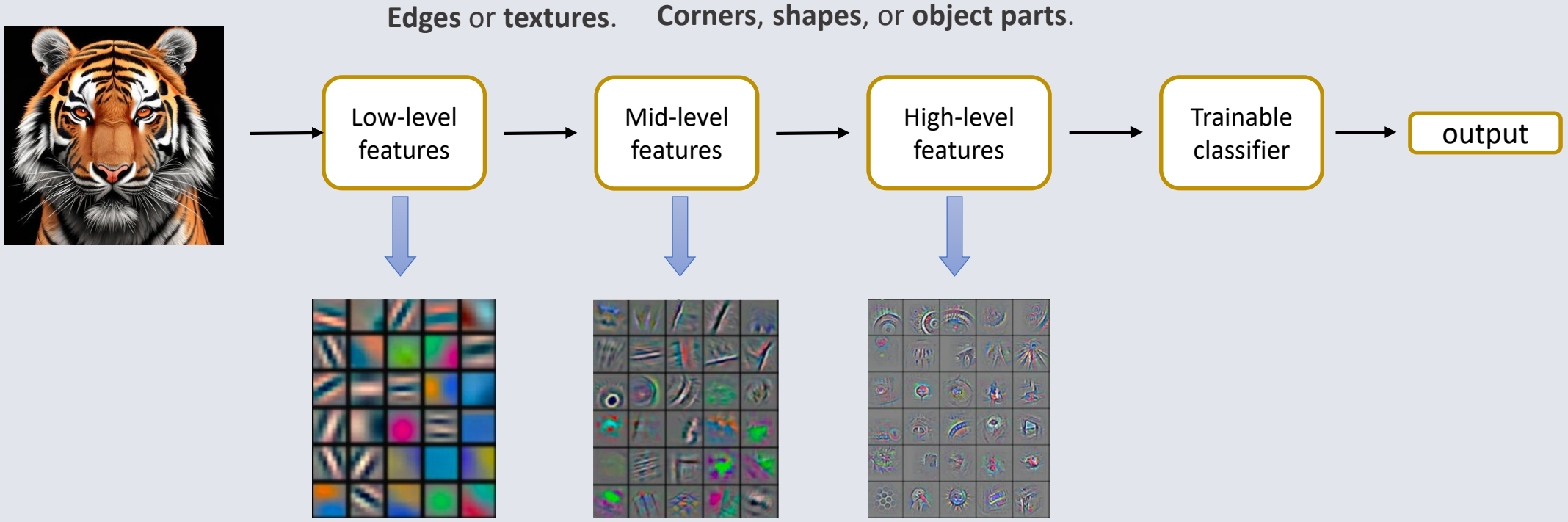
- Early hidden layers might detect low-level features like **edges**, **textures**, or **gradients**.
- Represented as a vector



Feedforward Deep Network: Components

Hidden Layer

- Early hidden layers might detect low-level features like **edges**, **textures**, or **gradients**.
- Represented as a vector



Feedforward Deep Network: Components

Hidden Layer

Edges or **textures**. **Corners**, **shapes**, or **object parts**. **Objects**

Low-level features → Mid-level features → High-level features → Trainable classifier → output

Increasing level of abstraction

Hierarchy of representations with increasing level of abstraction. Each stage is a kind of trainable nonlinear feature transform

Image recognition

Pixel → edge → texton → motif → part → object

Text

Character → word → word group → clause → sentence → story

Feedforward Deep Network: Components

Output Layer

The output layer produces the final prediction.

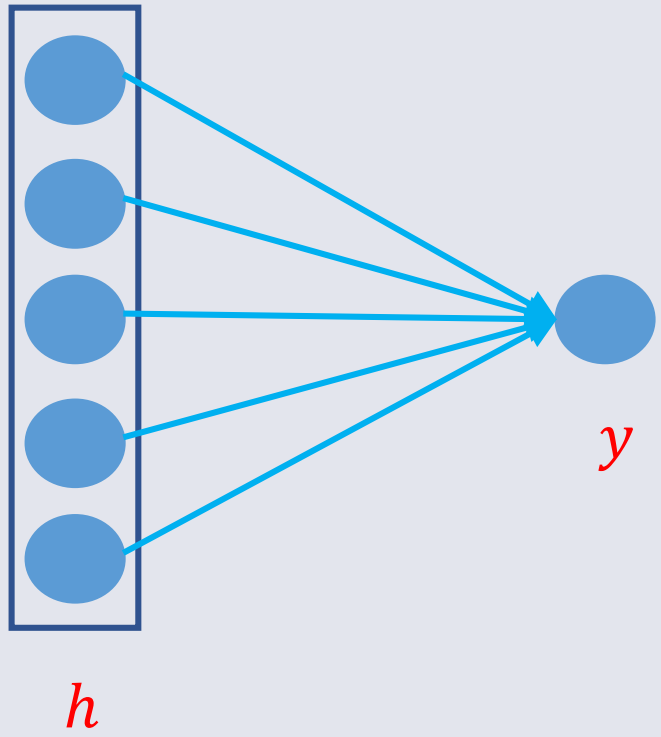
The number of neuron depends on the task.

Regression: 1 neuron with a linear activation function.

$y = w^T h + b$

Linear units: no nonlinearity

Example: Predict house prices.



Feedforward Deep Network: Components

Output Layer

The output layer produces the final prediction.

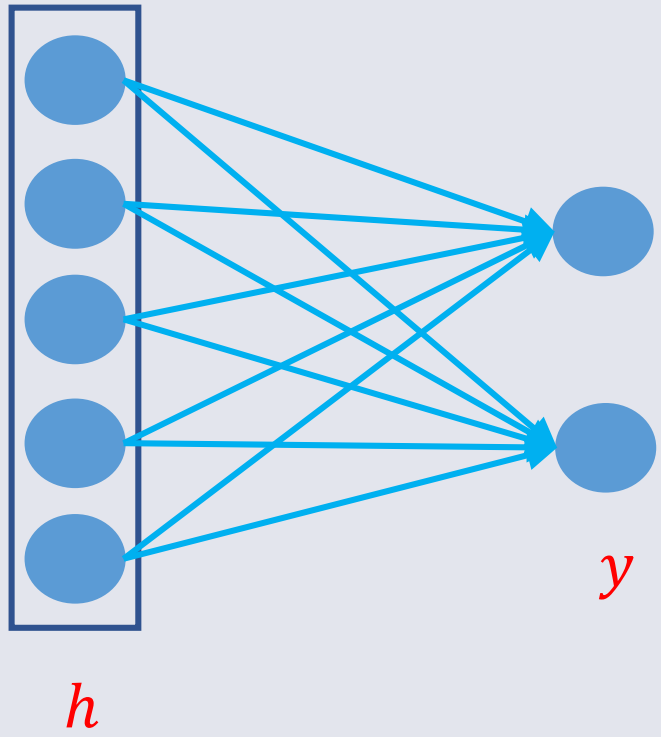
The number of neuron depends on the task.

Multi-Regression: n neuron with a linear activation function.

$y = w^T h + b$

Linear units: no nonlinearity

Example: Predict house prices.



Feedforward Deep Network: Components

Output Layer

The output layer produces the final prediction.

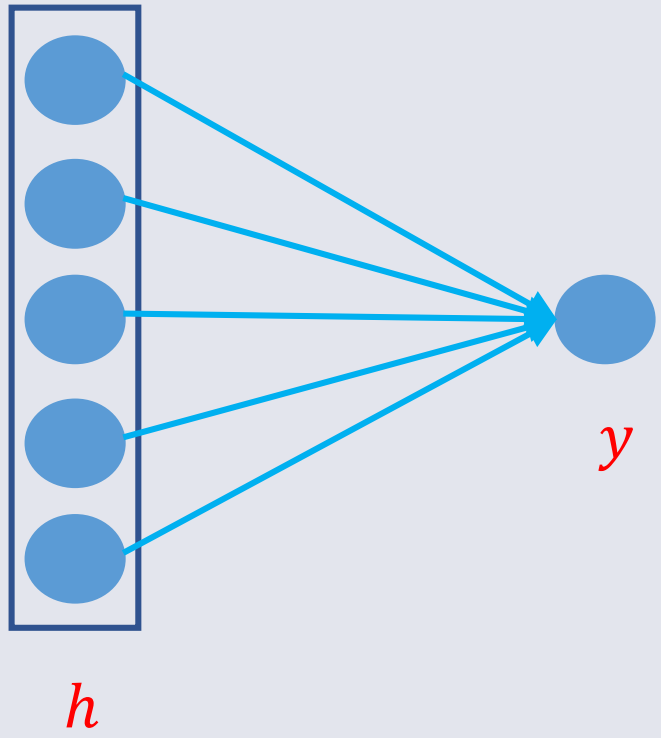
The number of neuron depends on the task.

Binary Classification: 1 neuron with a sigmoid activation function

$y = \sigma(w^T h + b)$

Linear units: no nonlinearity

Example: Approve loan (1) or deny loan (0).



Feedforward Deep Network: Components

Output Layer

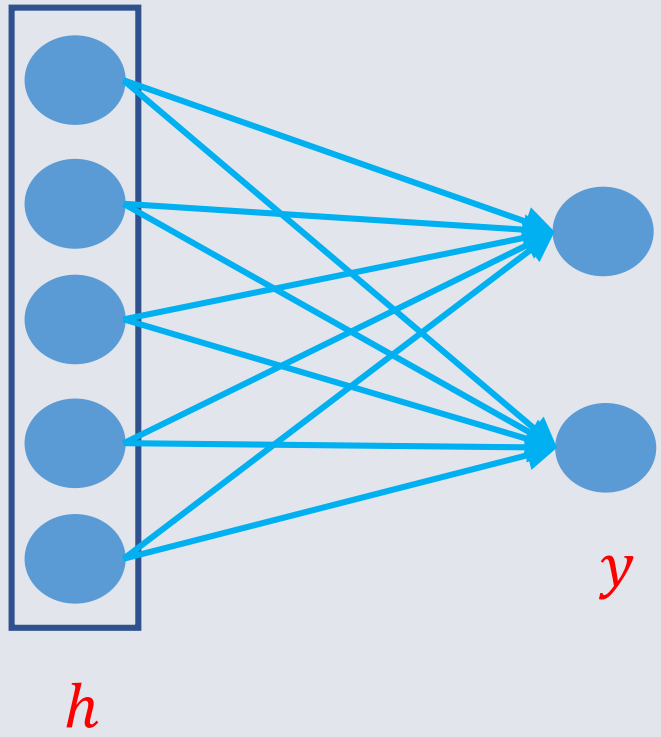
The output layer produces the final prediction.

The number of neuron depends on the task.

Multi-Class Classification: n neurons (one per class) with a softmax activation function.

y = softmax(z) where $z = W^T h + b$

Example: Classify handwritten digits (0, 1, 2, …9).



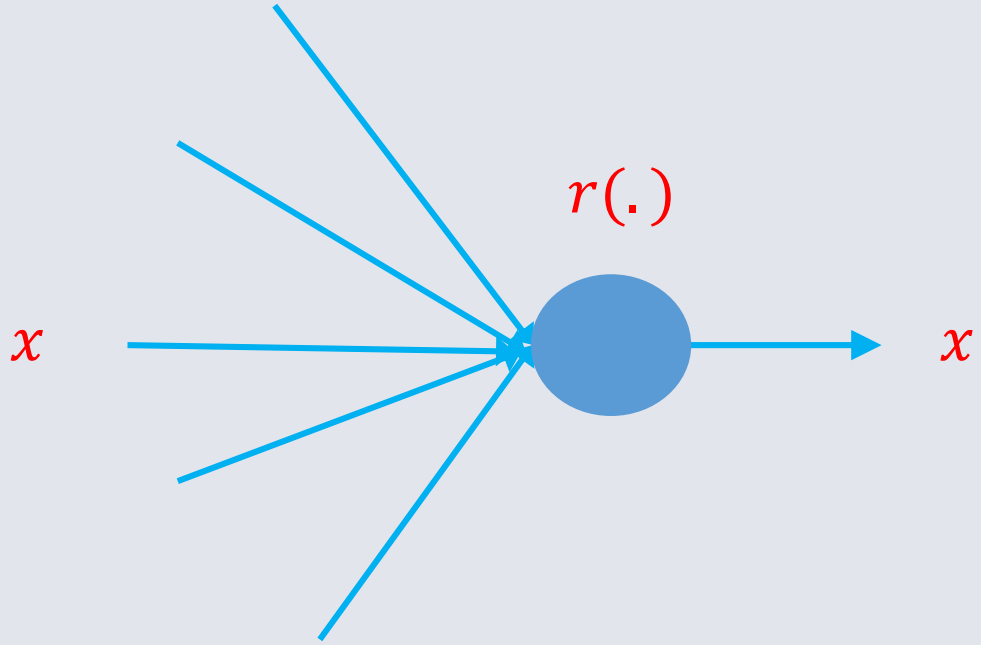
Feedforward Deep Network: Components

Activation Functions

- Consider a neuron.

$$Y = \sum(weight * input) + bias$$

- Now, the value of Y can be anything ranging from -inf to +inf.
- The neuron really doesn't know the bounds of the value.
- So how do we decide whether the neuron should fire or not?



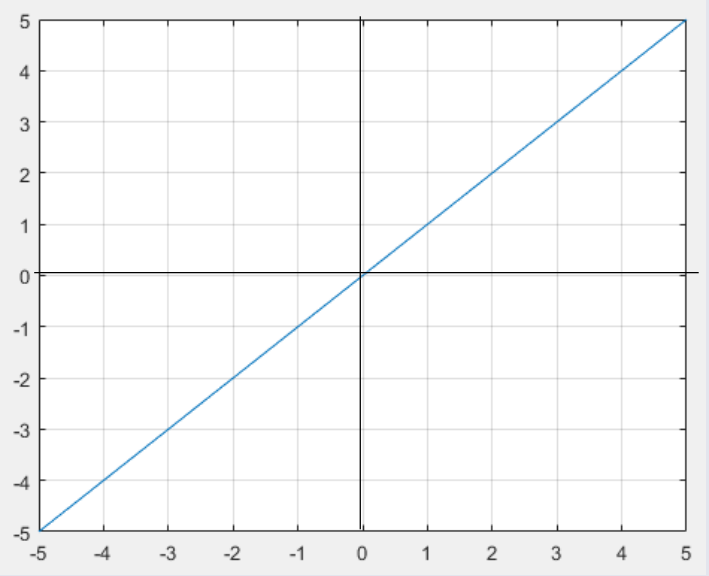
Feedforward Deep Network: Components

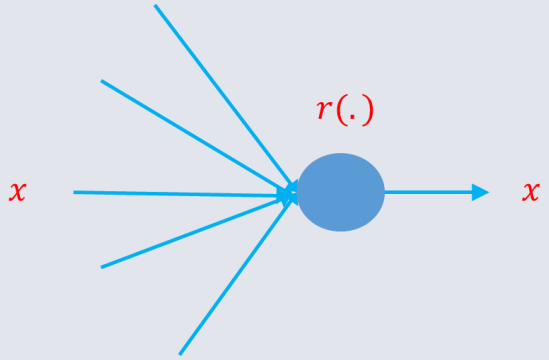
Activation Functions: **Linear**

- A straight line function where activation is proportional to input

$$f(x) = cx$$

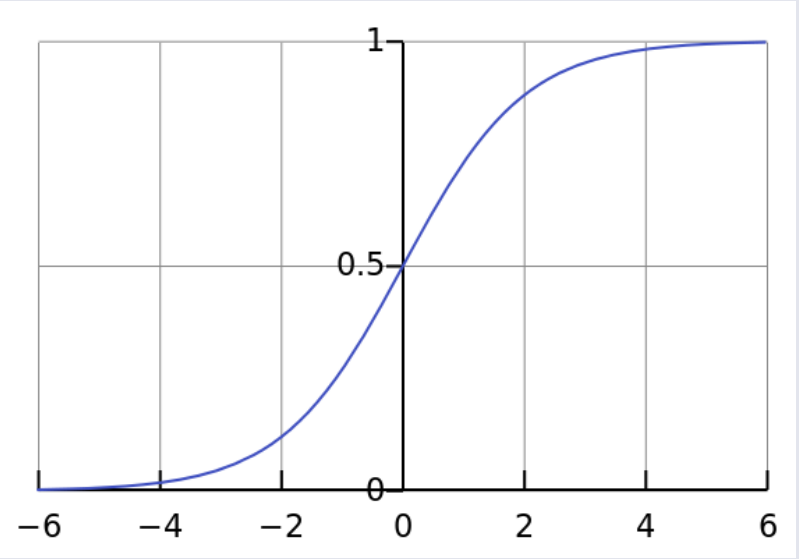
- Not appropriate for modeling complex nonlinear functions.





Feedforward Deep Network: Components

Activation Functions: **Sigmoid**



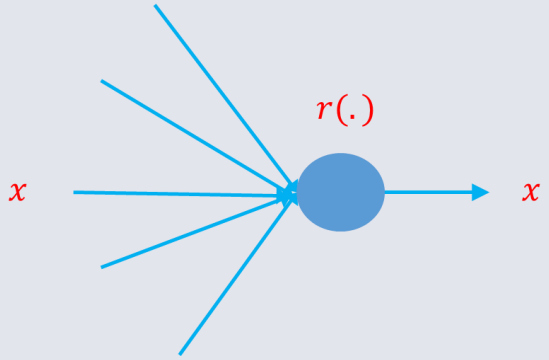
- If the value of Y is above a certain value, declare it “activated”.

$$f(x) = \frac{1}{1 + e^{-x}}$$

If sigmoid $(x) \geq 0.5$, predict class 1.

If sigmoid $(x) < 0.5$, predict class 0.

- Often used in binary classification tasks, where a threshold of 0.5 is used to make decisions.



Feedforward Deep Network: Components

Activation Functions: ReLu (Rectified Linear Unit)

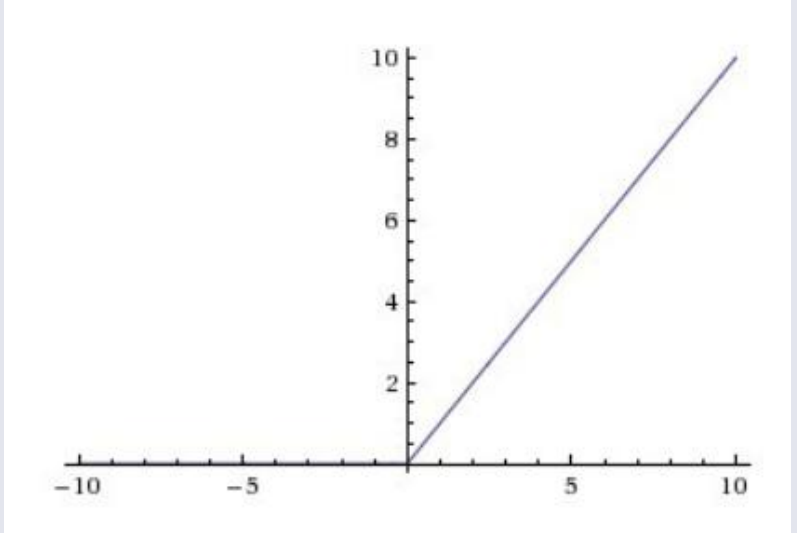
- It is a non-linear function and is capable of modeling complex functions.

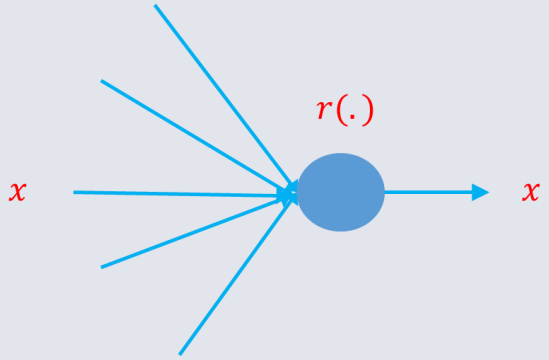
$$f(x) = \max(0, x)$$

  For any input x greater than 0, the output is x (no change).

  For any input x less than or equal to 0, the output is 0 (the neuron is "turned off").

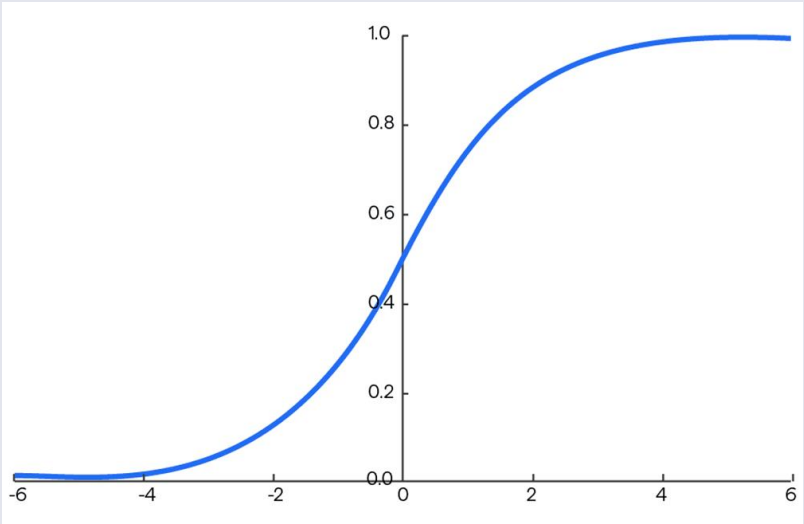
- The range of ReLu is [0, inf]
- It ensures sparsity of activations





Feedforward Deep Network: Components

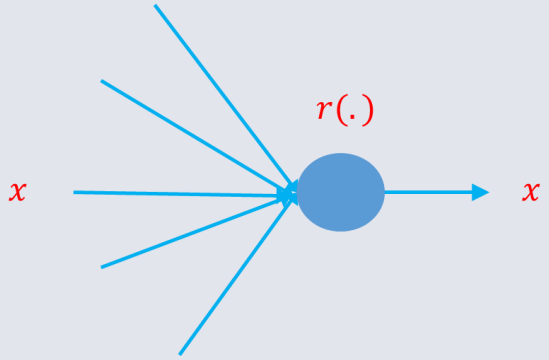
Activation Functions: **Softmax**



- In **softmax**, the output is a vector of probabilities, and there isn't a single threshold used in the same way as the sigmoid.
- **Thresholding** can be applied during **classification**:

$$Softmax\ (x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \max(0, x)$$

- For multi-class classification, after applying softmax, you typically select the class with the highest probability as the prediction.

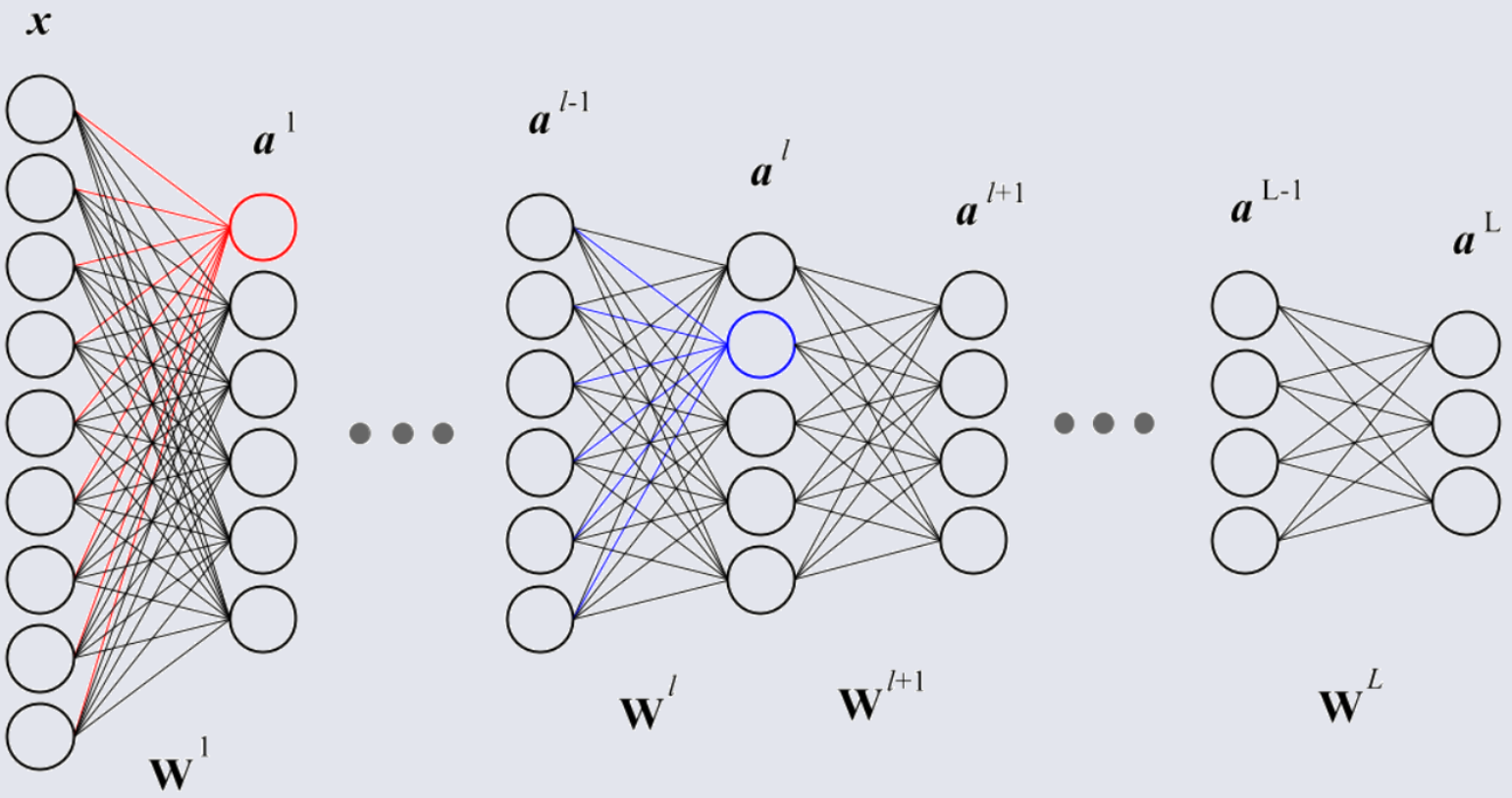


Forward Propagation

Problem:

MLPs have many weights and biases.

How do we update them to reduce the error?



Backpropagation

- **Solution**: Backpropagation calculates the gradients of the loss with respect to each weight and bias.
- Backpropagation is a common method for training a neural network.
- Optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs

Backpropagation

Steps:

1. Do forwards propagation

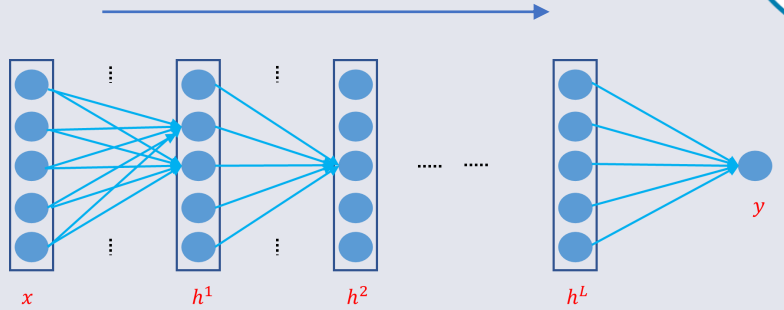
2. Compute the Loss/Error. (For instance, MSE)

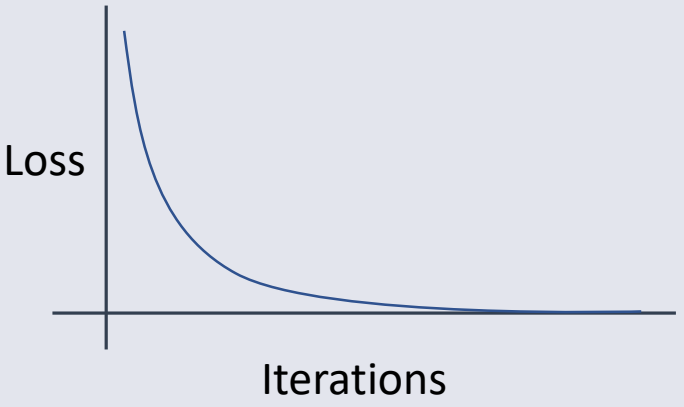
3. Backward Pass

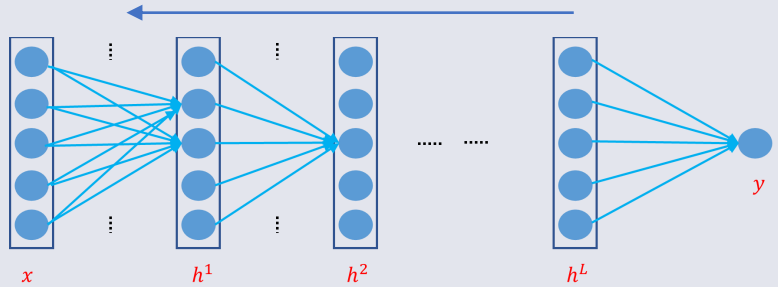
   Use the **chain rule** to compute gradients

   Adjust weights and biases

4. Repeat forward and backward passes to reduce the loss over time.





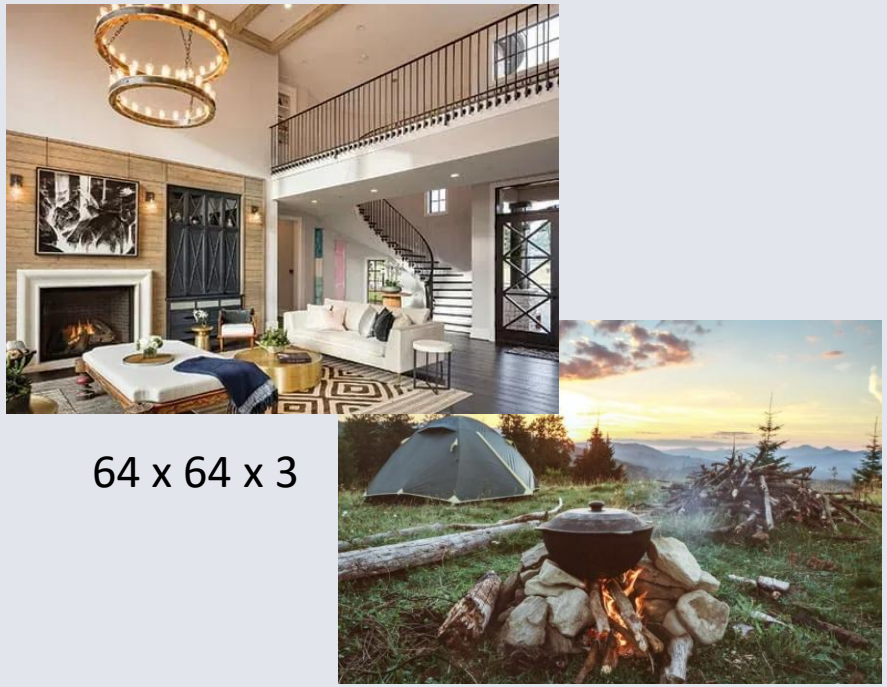


Backpropagation

What is the problem with Deep ANN?

- Consider an image classification problem (indoor/outdoor scene).
- Image size = 64 x 64 x 3 = 12288 (input layer)
- Suppose two hidden layers h1 = 1000 neurons, h2 = 500 neurons
- Outputs = 2 neurons (one for each class)
- How many parameters would we need to train??
- 12288 x 1000 + 1000 x 500 + 500 x 2 = 12789000

  (12.7 million) very small image and very small network



64 x 64 x 3

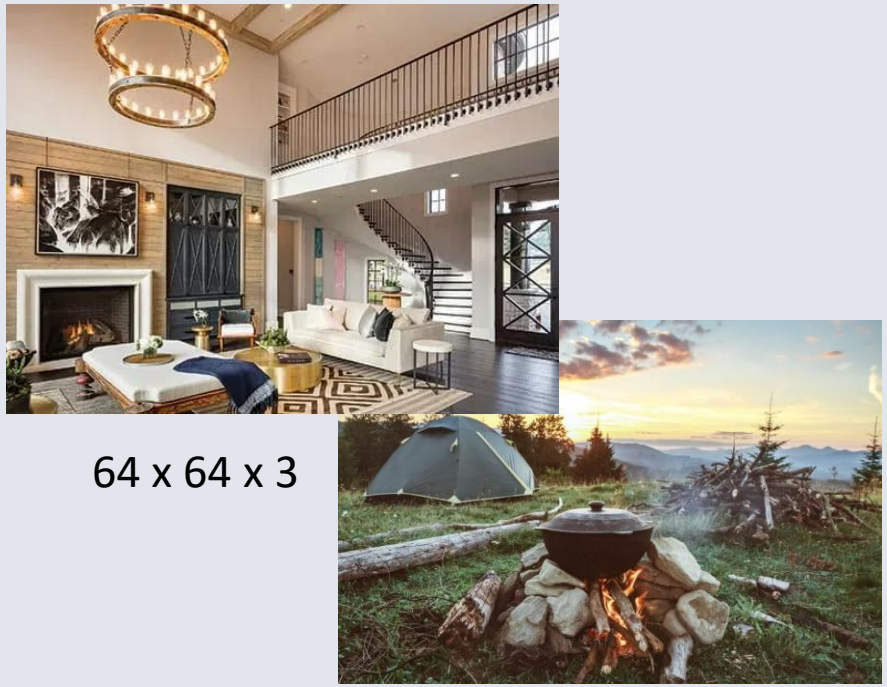
64 x 64 x 3

Backpropagation

What is the problem with Deep ANN?

- Consider an image classification problem (indoor/outdoor scene).
- Image size = 64 x 64 x 3 = 12288 (input layer)
- Suppose two hidden layers h1 = 1000 neurons, h2 = 500 neurons
- Outputs = 2 neurons (one for each class)
- How many parameters would we need to train??
- 12288 x 1000 + 1000 x 500 + 500 x 2 = 12789000

  (12.7 million) very small image and very small network



64 x 64 x 3

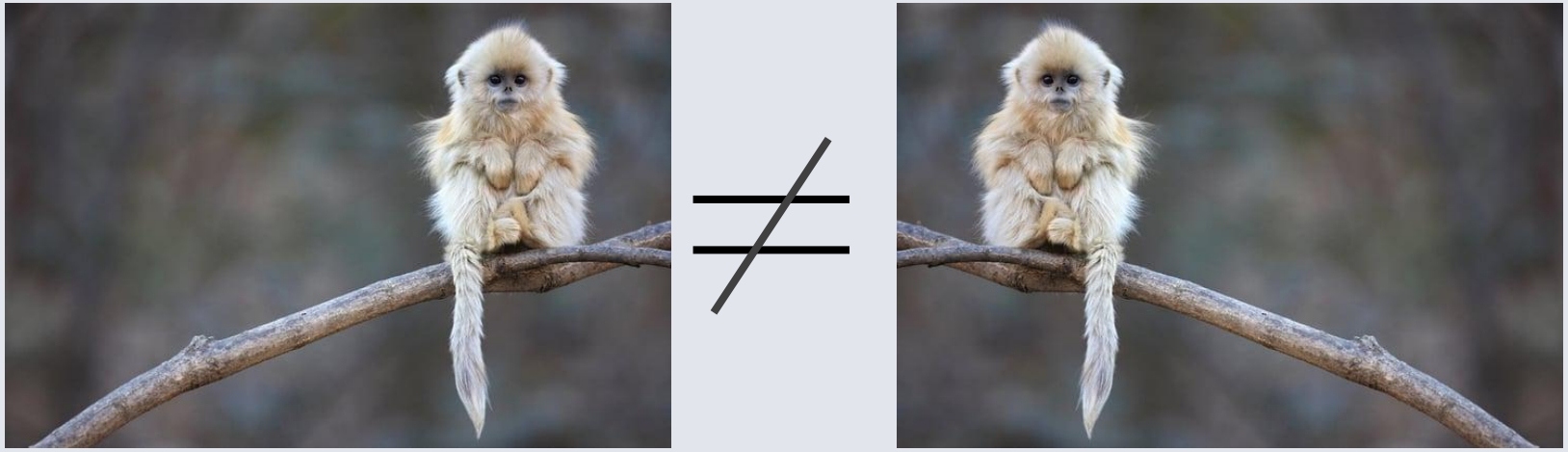
64 x 64 x 3

What if the image size = 256 x 256 x 3, and network has 4 layers with 1000 neurons and 1000 outputs?

Backpropagation

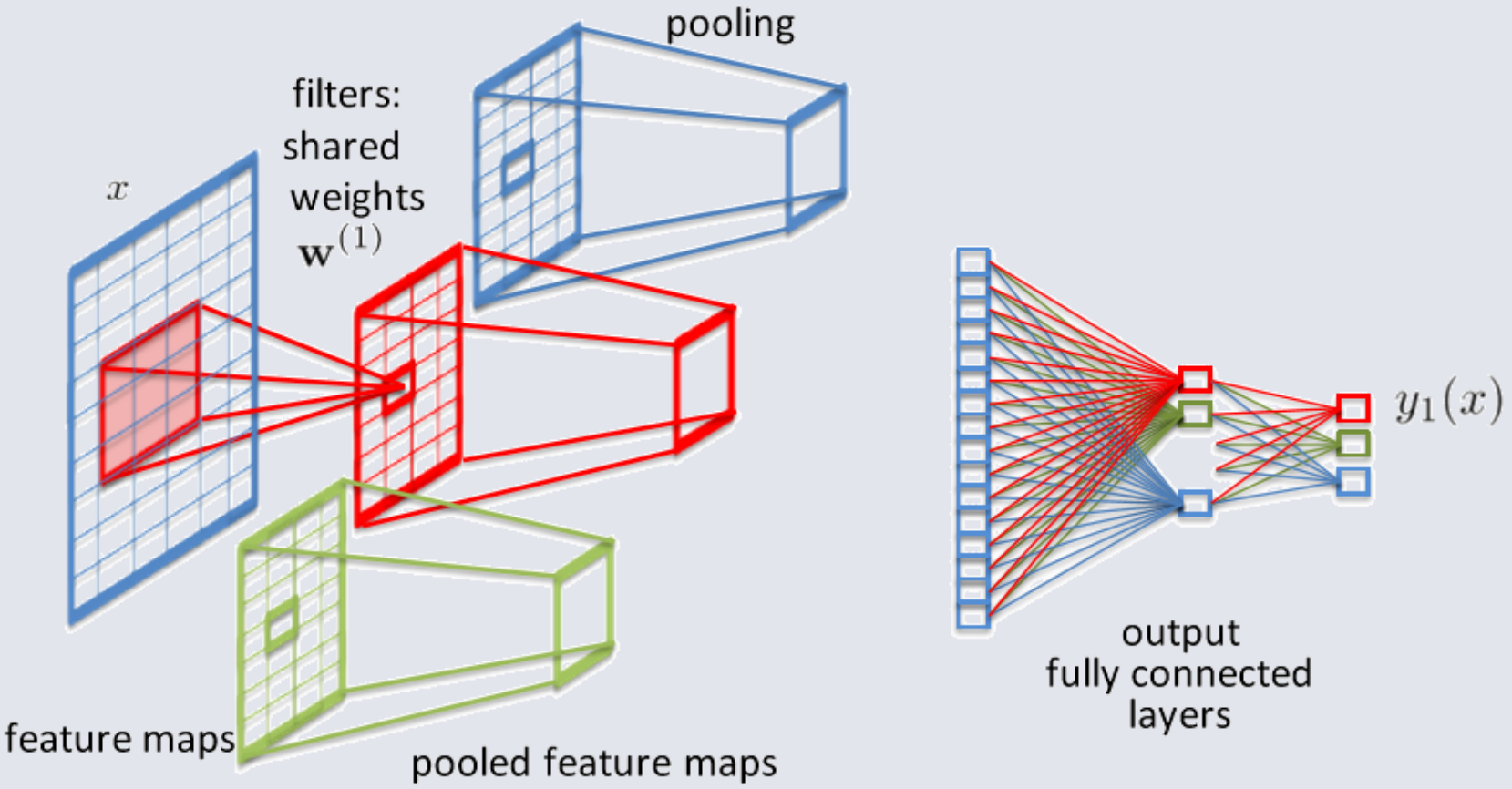
What is the problem with Deep ANN?

- Too Many Parameters:
- MLP[1000 1000 1000]-2000 000 parameters require too much training data.
- Computational and don’t generalize well.
- No spatial invariance for images.

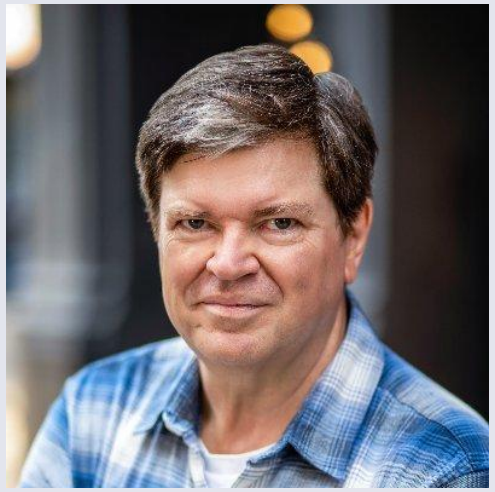


The Solution?

Convolutional neural networks (CNN)



Brief about CNN



Yann LeCun, Professor of Computer Science
The Courant Institute of Mathematical Sciences,
New York University
Room 1220, 715 Broadway, New York, NY 10003, USA.
(212)998-3283

Email: yann@cs.nyu.edu

In 1995, Yann LeCun and Yoshua Bengio introduced the concept of convolutional neural networks

Next week

Convolutional Neural Networks (CNNs) – Basics

Introduction to **CNNs**: Why **CNNs** are effective for image data.

Key components: Convolution layers, pooling layers, filters, stride, padding

Any Question?