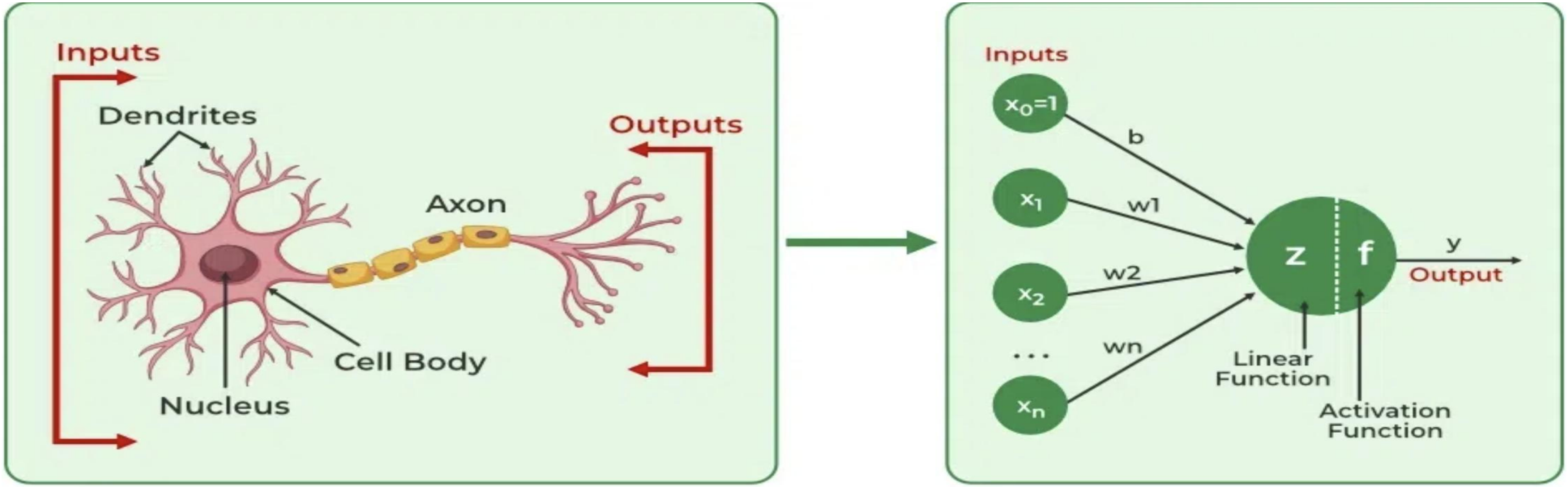
# Neural Network: Deep learning

# Neural networks

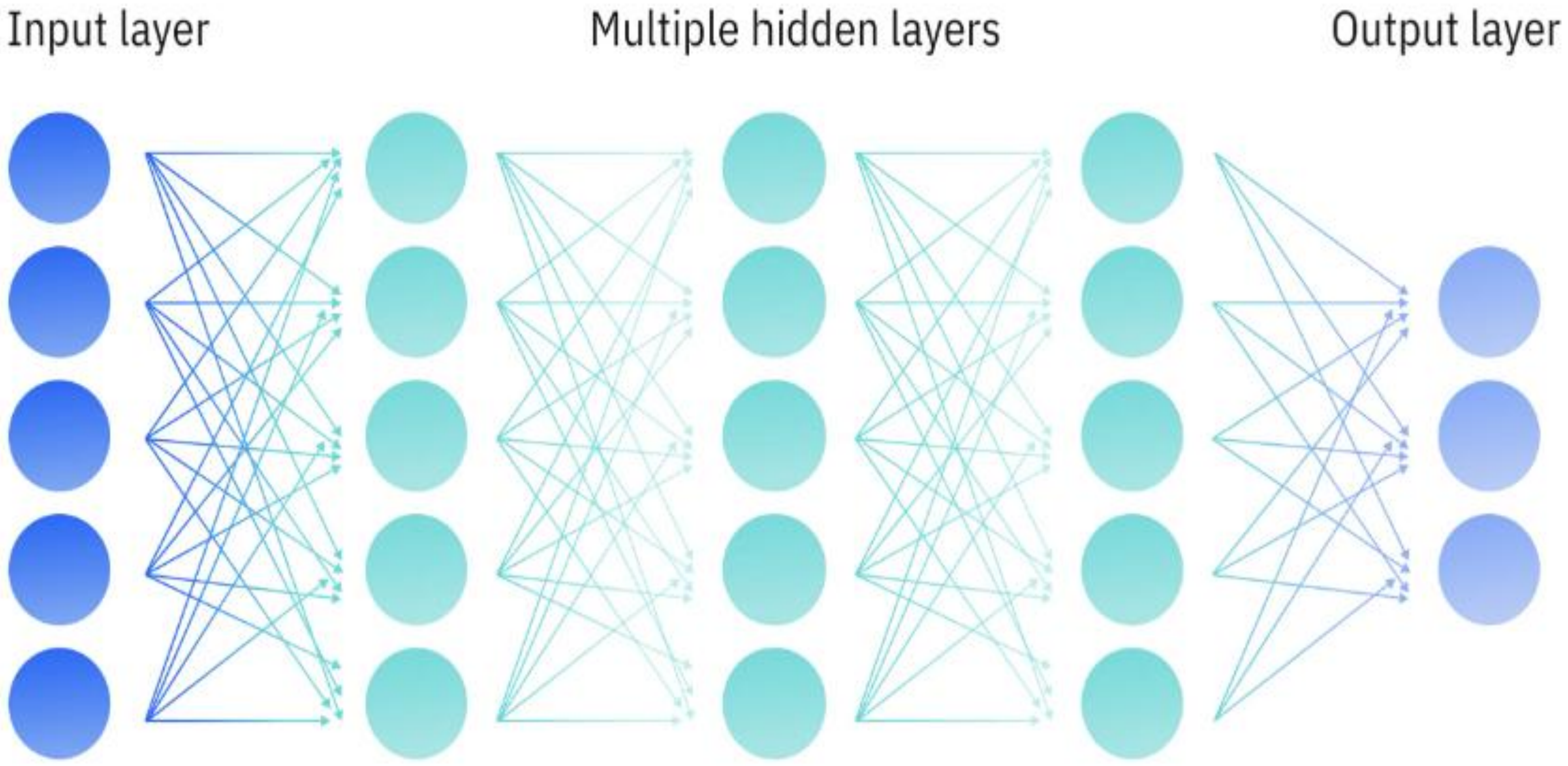
- Neural networks are machine learning models that mimic the complex functions of the human brain.
- These models consist of interconnected nodes (neurons) that process data, learn patterns and enable tasks such as pattern recognition and decision-making.
- A **neuron** (or **artificial neuron**) is the **basic computational unit** of a neural network that processes input data and produces an output.
- Deep learning is a subset of neural networks.
- All deep learning models are neural networks, but **not all neural networks are deep learning models**.

# Neural networks




*Analogy between a biological neuron and an artificial neuron*

# The components (layers) of a neural network



# The components (layers) of a neural network

1. Input layer:

- Receives the raw data
- Each neuron corresponds to one feature
- No computation is performed here
- Example:
  - ✓ Image → pixel values
  - ✓ Text → word embeddings
  - ✓ Sensor data → numerical readings

2. Hidden layer:

- One or more layers between input and output
- Responsible for feature learning (feature extraction)
- Each layer transforms data into a higher-level representation
- Example: In image recognition, one hidden layer might detect edges, the next might detect shapes, and a final one might recognize objects.

# The components (layers) of a neural network

## 3. Output Layer:

- Produces final prediction or result.
- The number of neurons in the output layer depends on the type of the problem
  - ✓ One neuron for a simple regression (predicting a single value) or binary classification (predicting two value)
    - ➢ Example: House price prediction (in case of simple regression), Spam vs Not Spam (in case of binary classification)
  - ✓ Multiple neurons (one for each class) for multi-class classification.
    - ➢ Example: Digit recognition (0–9) → 10 classes

      Emotion detection → Happy, Sad, Angry, Neutral

# Working of Neural Networks

1. Forward Propagation: When data is input into the network, it passes through the network in the forward direction, from the input layer through the hidden layers to the output layer.

   i. Linear Transformation: During this phase, each neuron in a layer receives inputs which are multiplied by the weights associated with the connections. These products are summed together and a bias is added to the sum, i.e.;

$$z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

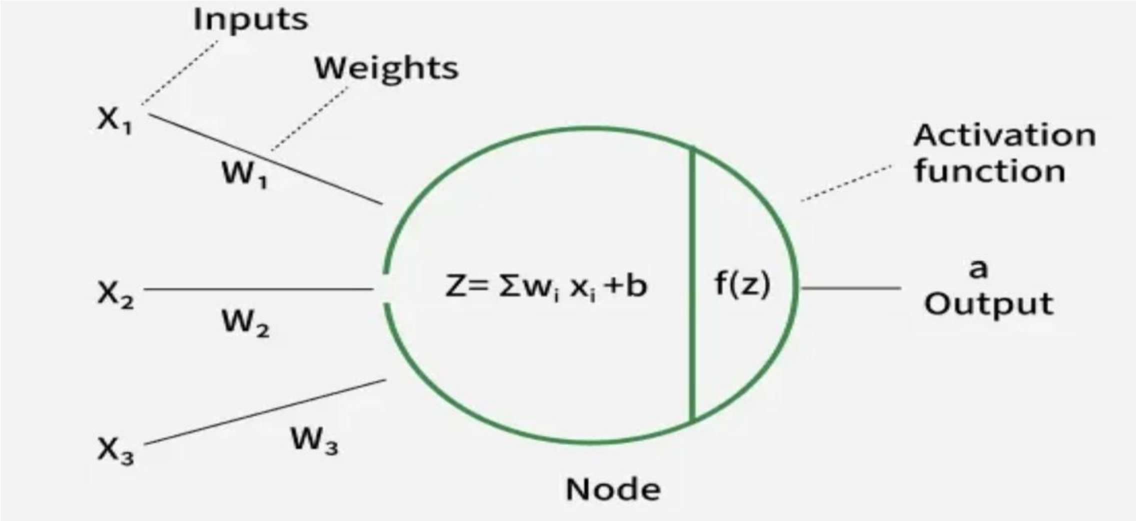
where, $w$ represents the weights, $x$ represents the inputs, $b$ is the bias

**Note:** Bias gives a neuron flexibility to make better decisions.
Bias allows a neural network to shift the activation function and learn patterns.

# Working of Neural Networks

ii. Activation Function: The result of the linear transformation (denoted as $z$) is then passed through an activation function.

- The activation function is crucial. It introduces non-linearity into the system, enabling the network to learn more complex patterns.




✓ Popular activation functions include ReLU, sigmoid, softmax and tanh.

✓ Output Range of different activation function:

ReLU-> 0 to ∞, sigmoid -> 0 to 1, softmax -> 0 to 1 (multi-class) , tanh -> -1 to 1

# Working of Neural Networks

## 2. Backpropagation:

- After forward propagation, the network evaluates its performance using a loss function.
  - ✓ Loss function measures the difference between the actual output and the predicted output.
  - ✓ The <u>goal of training is to minimize this loss</u>.

i. Loss Calculation: The network calculates the loss which provides a measure of error in the predictions.

- ✓ The loss function could vary.
- ✓ Common choices are mean squared error for regression tasks or cross-entropy loss for classification.

$$\text{Mean Squared Error} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- $n$ = number of data samples
- $y_i$ = actual (true) value
- $\hat{y}_i$ = predicted value

$$BCE = -\frac{1}{N}\Sigma_{i=1}^{N}(y_i.log(p_i) + (1 - y_i)log(1 - p_i))$$

Binary Cross-Entropy Loss

pi is the predicted probability that the sample belongs to class 1

# Working of Neural Networks

2. Backpropagation:

ii. Gradient Calculation: The network computes the gradients of the loss function with respect to each weight and bias in the network.

- ✓ This involves applying the chain rule of calculus to find out how much each part of the output error can be attributed to each weight and bias.

iii. Weight Update: Once the gradients are calculated, the weights and biases are updated using an optimization algorithm like stochastic gradient descent (SGD).

- ✓ The weights are adjusted in the opposite direction of the gradient to minimize the loss. The size of the step taken in each update is determined by the learning rate.

3. Iteration

- The process of forward propagation, loss calculation, backpropagation and weight update is repeated for many iterations over the dataset.
- This iterative process reduces the loss and the network's predictions become more accurate.

# Gradient

- A gradient is the rate of change of something.
- In deep learning, it tells us: How much the loss (error) changes if we change a weight a little bit.
- A gradient tells the network which direction to move (increase or decrease weights) to reduce the error.
  - ✓ Positive gradient → increasing $w$ increases loss
  - ✓ Negative gradient → increasing $w$ decreases loss
  - ✓ Zero gradient → weight is already optimal

# Example:

Email Content : Get free gift cards now!
Subject Line: Exclusive Offer
Objective: Is it spam or not

Step 1: Create a feature vector based on the analysis of keywords. The feature vector of the record can be presented as:

"free": Present (1)

"win": Absent (0)

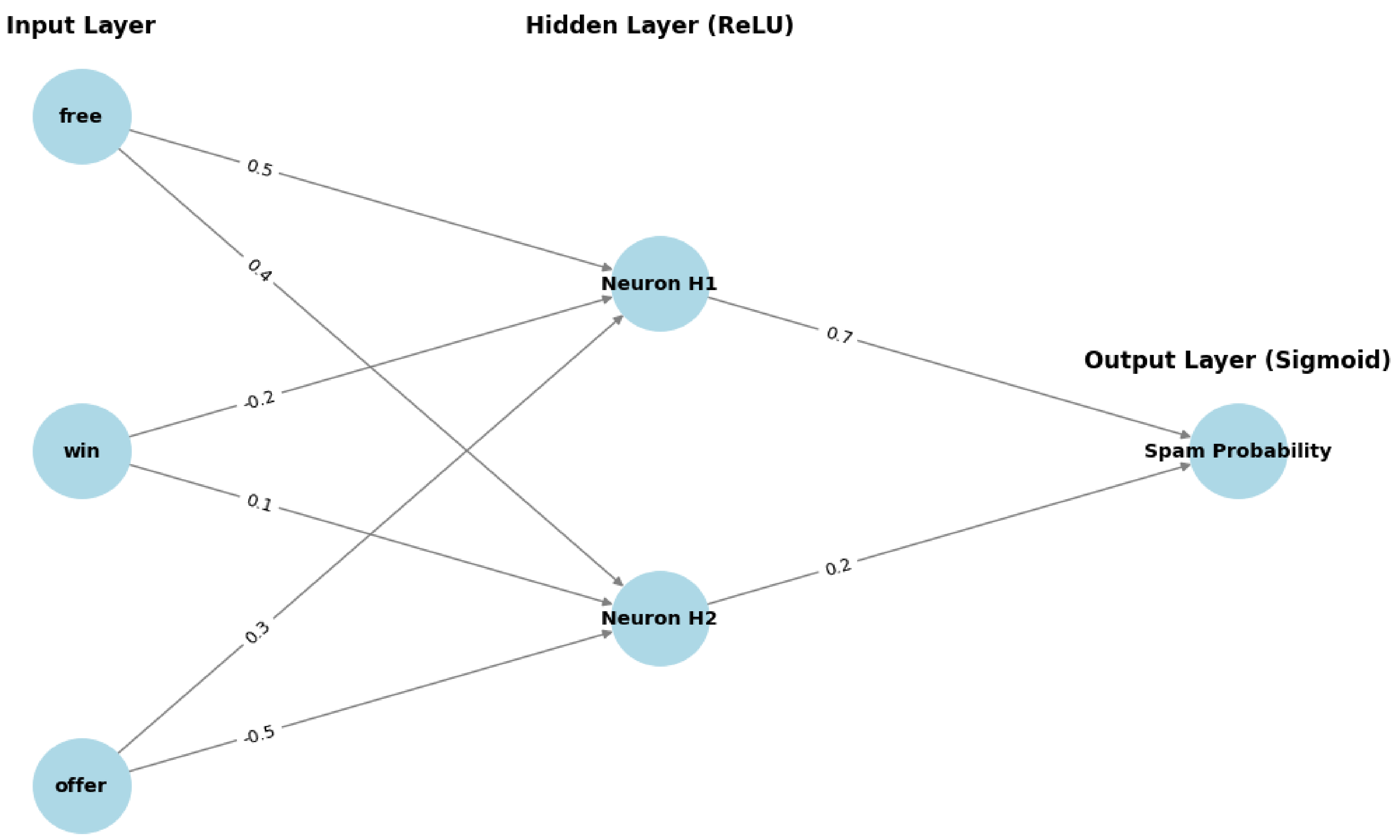
"offer": Present (1)

Step 2: The input layer contains 3 nodes that indicates the presence of each keyword.

Input Vector: **[1,0,1]**

**Note:** Vocabulary = {free, win, offer}

# Example:


Input Layer
Hidden Layer (ReLU)
Output Layer (Sigmoid)
free
win
offer
Neuron H1
Neuron H2
Spam Probability
0.5
0.4
-0.2
0.1
0.3
-0.5
0.7
0.2

# Example:

Step 3: The input vector is passed through the hidden layer

**Weights:**

Neuron H1: [0.5, −0.2, 0.3]

Neuron H2: [0.4, 0.1, −0.5]

Step 4: Weighted Sum Calculation

- **For H1: (**(1×0.5)+(0×−0.2)+(1×0.3)=0.5+0+0.3=0.8
- **For H2:** (1×0.4)+(0×0.1)+(1×−0.5)=0.4+0−0.5=−0.1

Step 5: Activation Function

If ReLu activation function is used, then

- **H1 Output:** ReLU(0.8)= 0.8
- **H2 Output:** ReLu(-0.1) = 0

# Example:

## Step 6: Output Layer

- **Output Weights:** [0.7, 0.2]
- **Input from Hidden Layer:** [0.8, 0]
- **Weighted Sum:** (0.8×0.7)+(0×0.2) = 0.56+0 = 0.56
- **Activation (Sigmoid):** $\sigma(0.56) = \frac{1}{1+e^{-0.56}} \approx 0.636$

## Step 7: Final Classification

Since this value is greater than 0.5, the neural network classifies the email as spam (1).

# Deep Learning

- Deep learning is an artificial intelligence (AI) method that teaches computers to process data in a way inspired by the human brain.

- Deep learning is a specialized form of machine learning that uses multi-layered neural networks (hence Deep) to analyze.

- Deep learning models can recognize complex pictures, text, sounds, and other data patterns to produce accurate insights and predictions.

- Deep learning methods can be used to automate tasks that typically require human intelligence, such as describing images or transcribing a sound file into text.

# Use cases of deep learning?

- Self-driving cars use deep learning models for object detection.
- Defense systems use deep learning to flag areas of interest in satellite images.
- Medical image analysis uses deep learning to detect cancer cells for medical diagnosis.
- Factories use deep learning applications to detect when people or objects are within an unsafe distance of machines.

# How does deep learning work?

- Deep learning models are neural networks designed after the human brain.

- A human brain contains millions of interconnected biological neurons that work together to learn and process information. Similarly, artificial neurons are software modules called nodes that use mathematical calculations to process data.

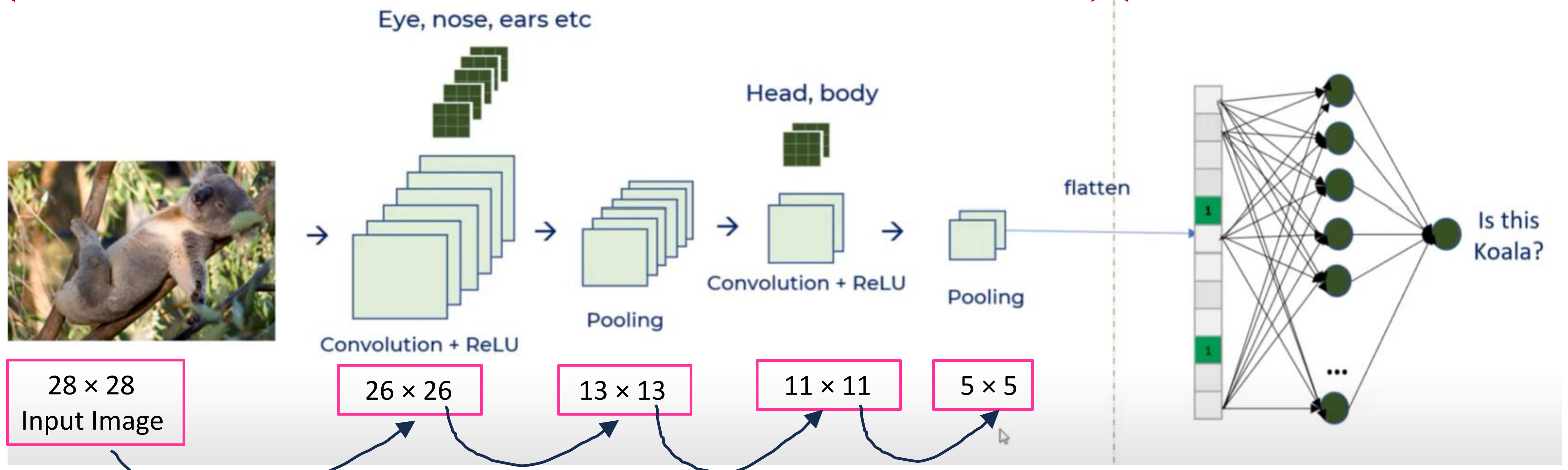
- Deep learning neural networks, or artificial neural networks, comprise many layers of artificial neurons that work together to solve complex problems.

# Convolutional Neural Network (CNN)

CNN is a deep learning model used for image classification, detection, and segmentation.

Feature Extraction/Feature Learning

Classification




**3 × 3**
**Kernal or filter**

**2 × 2**
**Polling**

**3 × 3**
**Kernal or filter**

**2 × 2**
**Polling**

**Output = [(I – K)/S] + 1**

Here stride (S) is 1 and number of channel is also 1

- Each pixel stores a value
- For b/w image, channel = 1
- For RGB, channel = 3

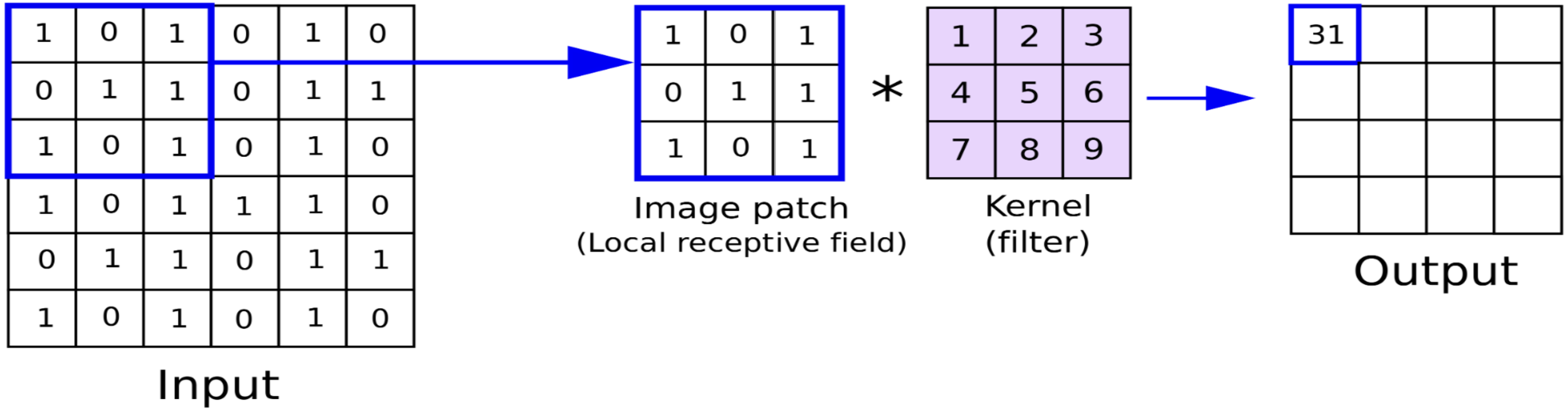
# CNN

## 1. Feature Extraction/Learning:

- Detecting all the feature like ear, eyes, nose etc.

### I. Convolution:

- Kernal can be used to extract the feature
- A kernel is a small matrix (e.g., $3 \times 3$, $5 \times 5$ ) that slides across an image, performing element-wise multiplication and summing up the results to produce a single output value.



# CNN

II. ReLu (Rectified Linear Unit) activation function:

- ReLu activation function takes your feature value: Relu = max (0, x)
- If the value is negative, it will replace it with 0. If it is more than zero it will be keep it same
- It makes the model to non-linear (Real-world data is not linear)
- It also speed up the training

III. Polling layer: It is used to reduce the size of an image, i.e., it reduce the dimensions

- The max polling:
- Average polling

# The max polling:

- Window of 2 by is taken

| | | | |
|---|---|---|---|
| 5 | 1 | 3 | 4 |
| 8 | 2 | 9 | 2 |
| 1 | 3 | 0 | 1 |
| 2 | 2 | 2 | 0 |

| | |
|---|---|
| 8 | **9** |
| 3 | 2 |

2 by 2 filter with stride = 2

**4 By 4 is reduced to 2 by 2**

# The max polling:

| | | | |
|---|---|---|---|
| 5 | 1 | 3 | 4 |
| 8 | 2 | 9 | 2 |
| 1 | 3 | 0 | 1 |
| 2 | 2 | 2 | 0 |

| | |
|---|---|
| 8 | 9 |
| 3 | 2 |

2 by 2 filter with stride = 2

**4 By 4 is reduced to 2 by 2**

# The max polling:

| | | |
|---|---|---|
| 0 | **1** | 0 |
| 0 | 0.11 | 0 |
| 0 | 0.33 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |

| | |
|---|---|
| 1 | **1** |
| 0.33 | 0.33 |
| 0.33 | 0.33 |
| 0 | 0 |

2 by 2 filter with stride = 1

# The max polling:

| | | |
|---|---|---|
| 0 | **1** | 0 |
| 0 | 0.11 | 0 |
| 0 | 0.33 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |

| | |
|---|---|
| 1 | **1** |
| 0.33 | 0.33 |
| 0.33 | 0.33 |
| 0 | 0 |

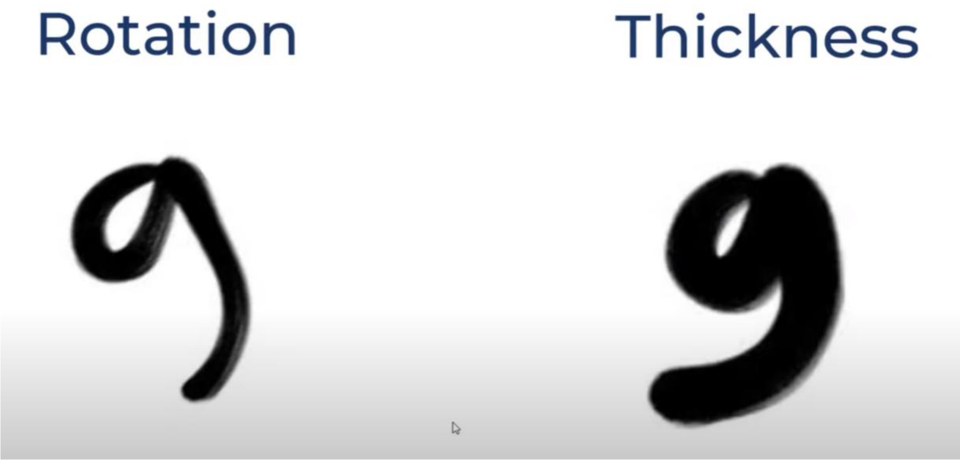
2 by 2 filter with stride = 1

# CNN

- Average polling is same as max polling, we take the average in place of maximum
- **Benefits of polling:**
  - Reduce dimension and computation
  - Reduce overfitting because there are less parameter
    - ➢ Overfitting or high variance occurs when the accuracy of your training dataset is greater than your testing accuracy.

# CNN

- CNN can not handle rotation and scale by itself




- Due to this, training dataset should have rotated and scaled sample
- If it does not have than pick some of the sample from training data set and rotate and scale them using Data Augmentation or Spatial Transformer Networks (STN)

# CNN: Feedforward Propagation

**Feed Forward (Forward Pass):** It passes the input data through the neural network layer by layer to produce an output.

- CNN forward pass helps the model to understand the image and produce an output.

  Image → Conv → ReLU → Pool → Fully Connected Layer → Output

- The image passes through:
  - Convolution layers (detect patterns like edges)
  - ReLU (activation)
  - Pooling (reduce size)
  - Fully connected layers (classify)
- The output is a **prediction.**

# CNN: Backward Propagation

**Backward Propagation (Backward Pass):**

- After comparing the prediction with the true label, we calculate the loss.
- The loss is sent backward through the network.
- Backpropagation helps the CNN learn better filters (e.g., to detect objects more accurately).

# CNN: Training

**1. Forward Pass:**

Input Image → Conv → ReLU → Pool → FC → Output

**2. Compute Loss:**

Compare Prediction vs True Label

**3. Backward Pass:**

Backpropagate Loss → Compute Gradients

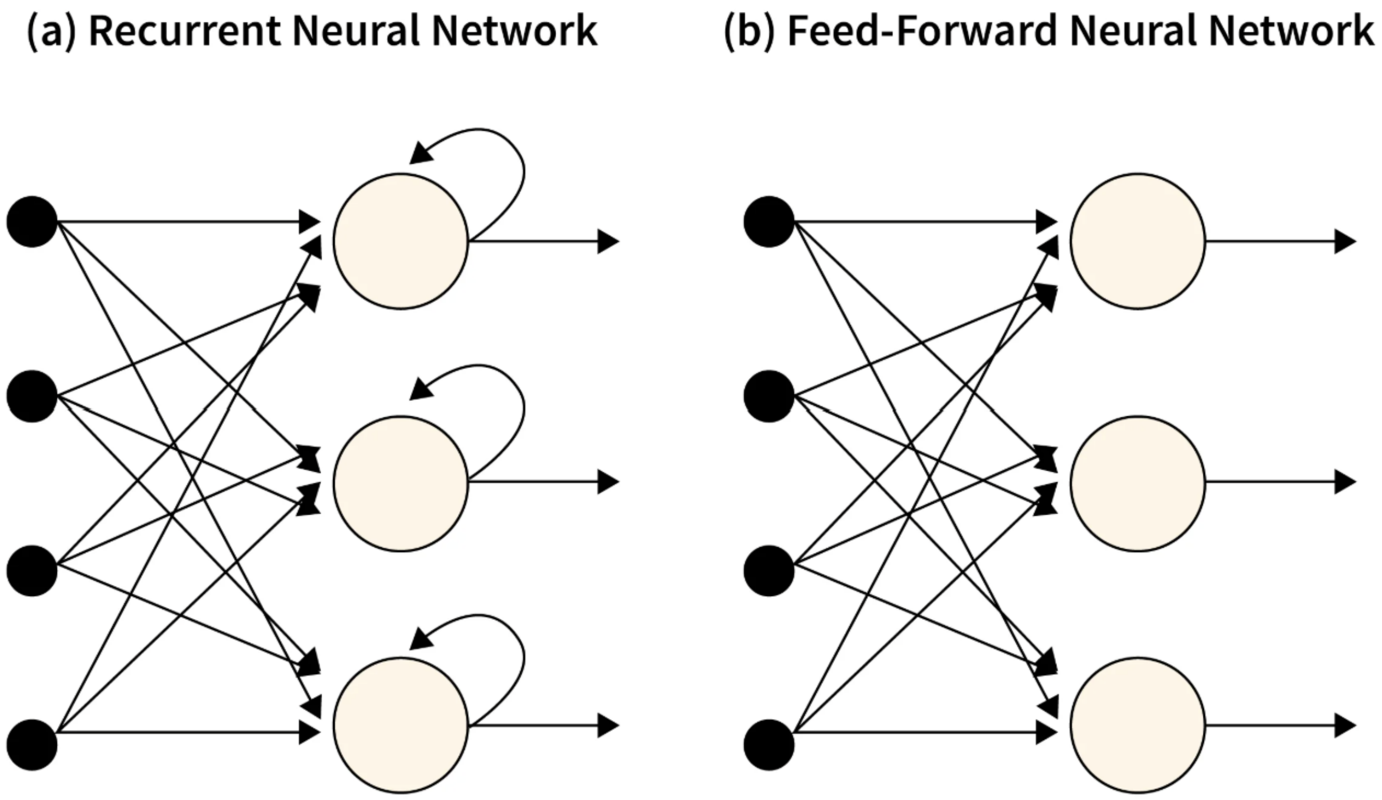
**4. Update Weights:**

Use Optimizer to adjust filter weights

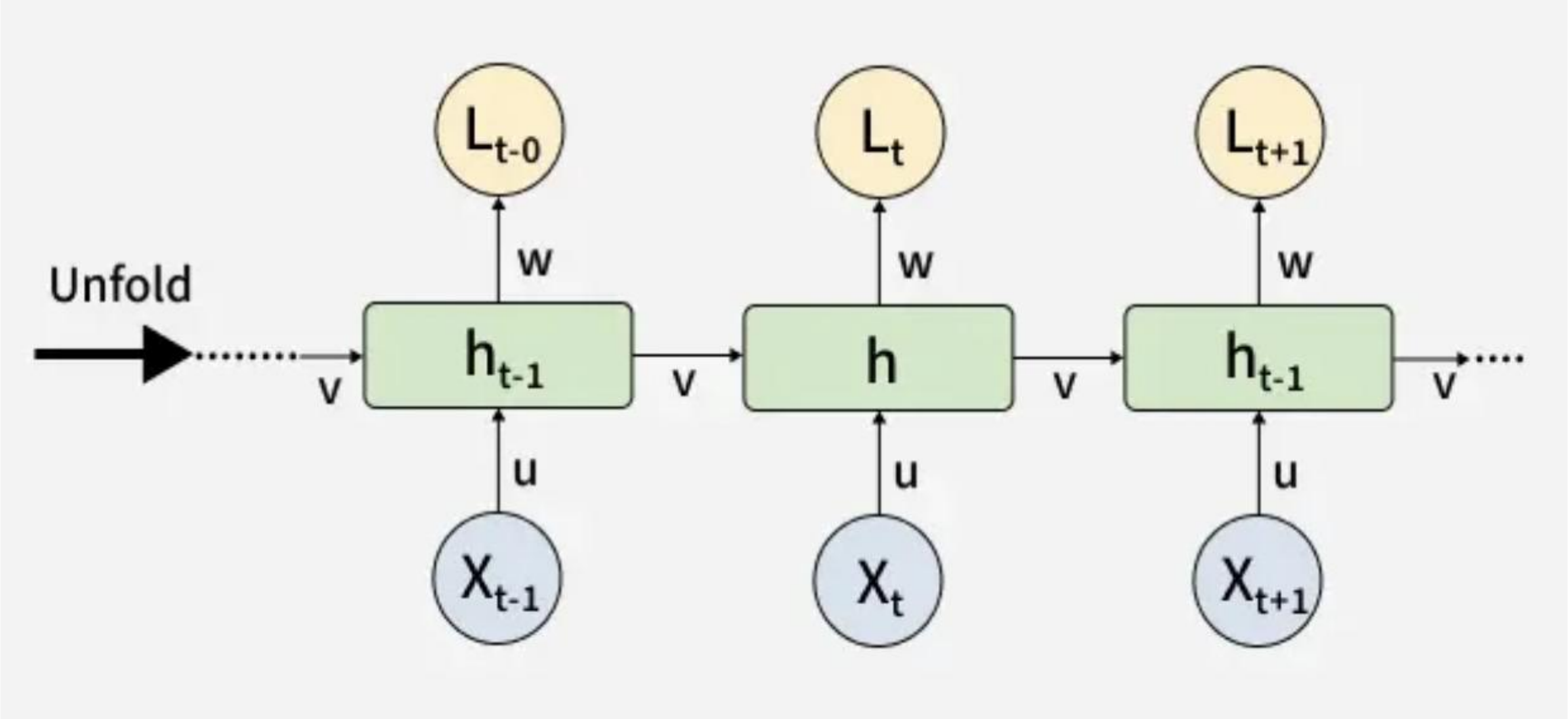
# Recurrent Neural Network

- Recurrent Neural Networks (RNNs) are a specific type of Neural Networks that are especially relevant for sequential data like time series, text, or audio data.
- Traditional neural networks process each input independently, meaning they cannot retain information about previous inputs. This makes them ineffective for tasks that require understanding sequences, such as time series forecasting or natural language processing.
- RNNs however, process the data sequentially, which enables them to remember data from the past.
- An RNN uses the output of one step as input of the next step in addition to the input data and in that way creates a connection and a memory to data of previous steps.

# Recurrent Neural Network



# Recurrent Neural Network: Unfold



# Type of Recurrent Neural Network

- **One-to-One RNN:** Receives a single input and produce a single output
  - ✓ Example: It is used for straightforward classification tasks such as binary classification where no sequential data is involved.
- **One-to-Many RNN:** Receives a single input and produce multiple outputs over time
  - ✓ Example: In image captioning a single image can be used as input to generate a sequence of words as a caption.

# Type of Recurrent Neural Network

- **Many-to-One RNN:** Receives a sequence of inputs and generates a single output.
  - ✓ Example: In sentiment analysis the model receives a sequence of words (like a sentence) and produces a single output like positive, negative or neutral.
- **Many-to-Many RNN:** Receives a sequence of inputs and generates a sequence of outputs
  - ✓ Example: In language translation task a sequence of words in one language is given as input and a corresponding sequence in another language is generated as output.