Introduction to Deep Learning

Sammie Omranian
CIberCATSS 2024

6/13/2024

Agenda

- What is Deep Learning?
- What is a Neural Network?
- Structure of Neural Networks
- How Neurons Work
- Training a Neural Network
- How Neural Networks Learn
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)

What is a Neural Network?

- Neural networks are computational models inspired by the human brain, used to recognize patterns and make decisions.

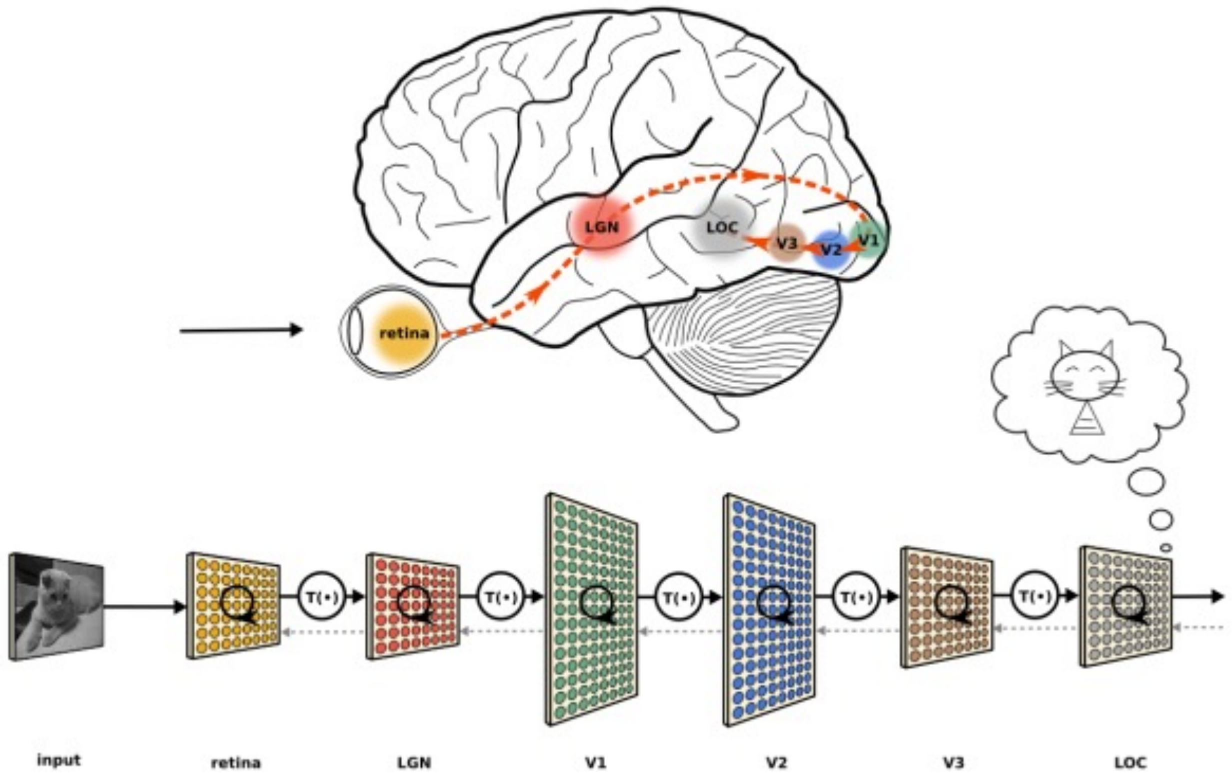


Image credit

What is Deep Learning?

- Deep learning is a subset of machine learning where neural networks with many layers learn from large amounts of data.
- **Purpose:** To enable machines to automatically learn complex patterns and representations from data.

- Neural networks vs. deep learning
- The word “deep” in deep learning is referring to the depth of layers in a neural network.
- A neural network that consists of more than three layers—which would be inclusive of the inputs and the output—can be considered a deep learning algorithm.
- A neural network that only has two or three layers is just a basic neural network.

Structure of Neural Networks

- Layers: Input layer, hidden layers, output layer.
- Neurons: Basic units of neural networks.
- Connections: Weighted connections between neurons.

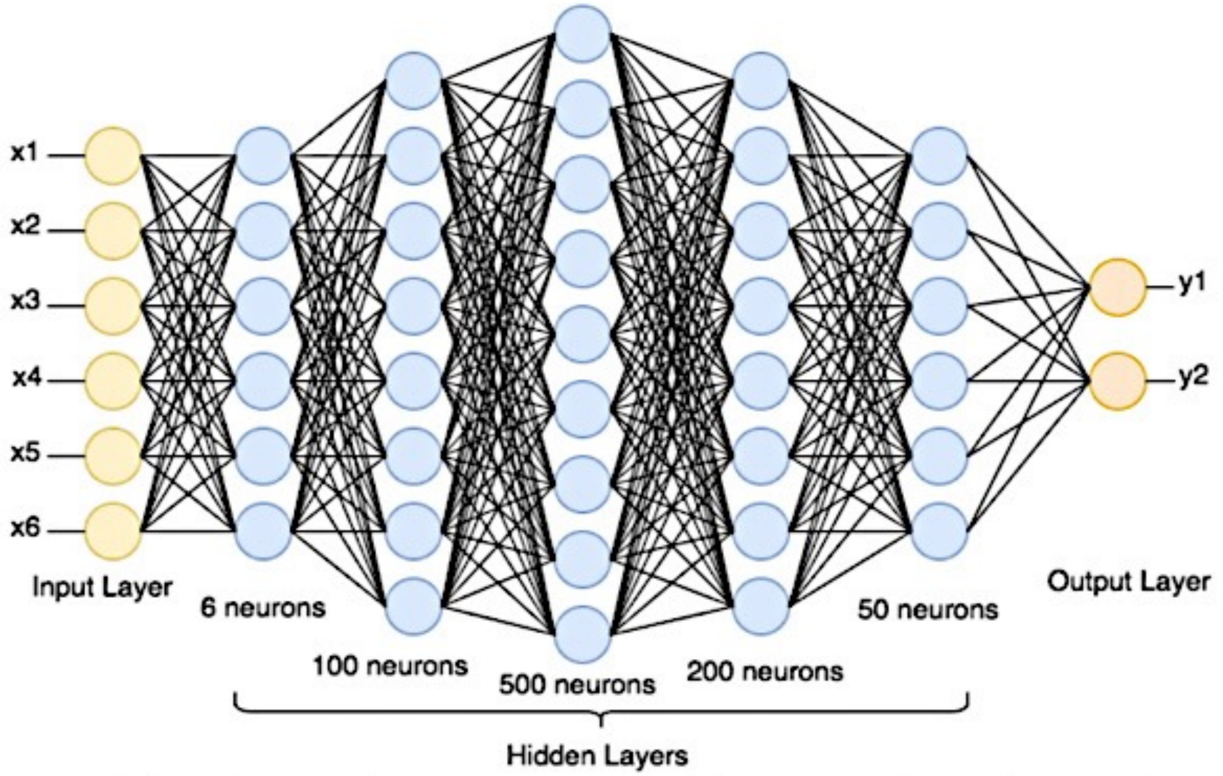


Image credit

How Neurons Work

- Activation: Neurons receive inputs, apply weights, sum them up, and pass the result through an activation function.
- Activation Functions: Common functions include ReLU, Sigmoid, Softmax, and Tanh.
- Threshold: Neurons are activated if the output exceeds a certain threshold.

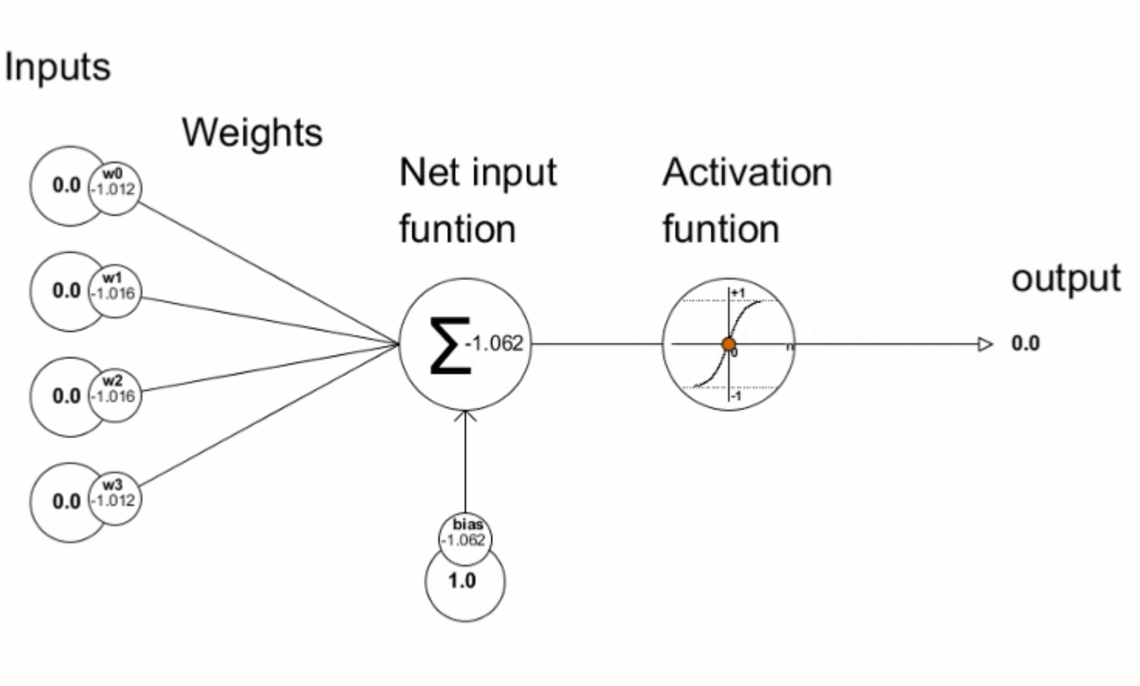


Image credit

Training a Neural Network

- Training a neural network primarily involves two phases: **forward propagation** and **backpropagation**.

- Forward Propagation: Process of passing inputs through the network to get an output.

- Backpropagation: process for adjusting weights to minimize the loss function using gradient descent.

Training a Neural Network

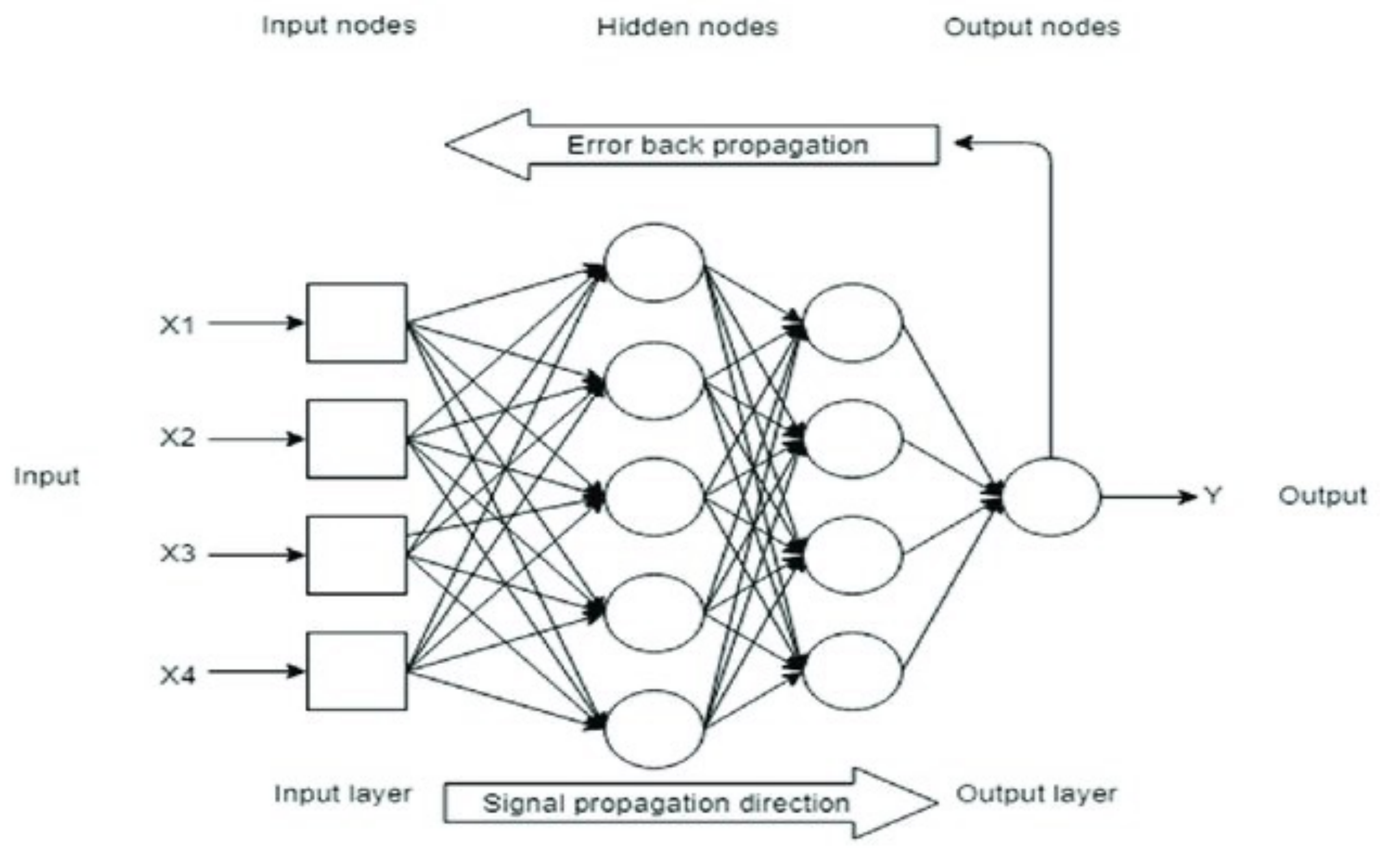


Image credit

Training a Neural Network

How NN works?

1. Forward Propagation:
   - The network takes an input and computes the predicted output.

2. Loss Calculation:
   - calculated after forward propagation
3. Backpropagation:
   - Using the error calculated from the loss function, the network computes gradients with respect to each weight and bias.
   - These gradients are then used to update the weights and biases to reduce the error in future predictions.

Training a Neural Network

How NN works?

4. Loss Calculation:

- The loss function is used to calculate the error between the predicted output and the actual target values.

- This step happens after forward propagation and before backpropagation.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

y_i is the actual target value for the i-th sample.

$\hat{y}_i$ is the predicted value for the i-th sample.

n is the number of samples.

- The calculated loss value is used to quantify how well the network is performing.

Training a Neural Network

How NN works?

5. Backpropagation Steps:

- Output Layer Gradient:
  - Calculate the gradient of the loss with respect to the output.
- Hidden Layer Gradients:
  - Propagate the gradient back through the network to calculate the gradients with respect to the hidden layers.
- Weight and Bias Updates:
  - Update the weights and biases using the computed gradients and a learning rate.

Training a Neural Network

More details for Forward Propagation:

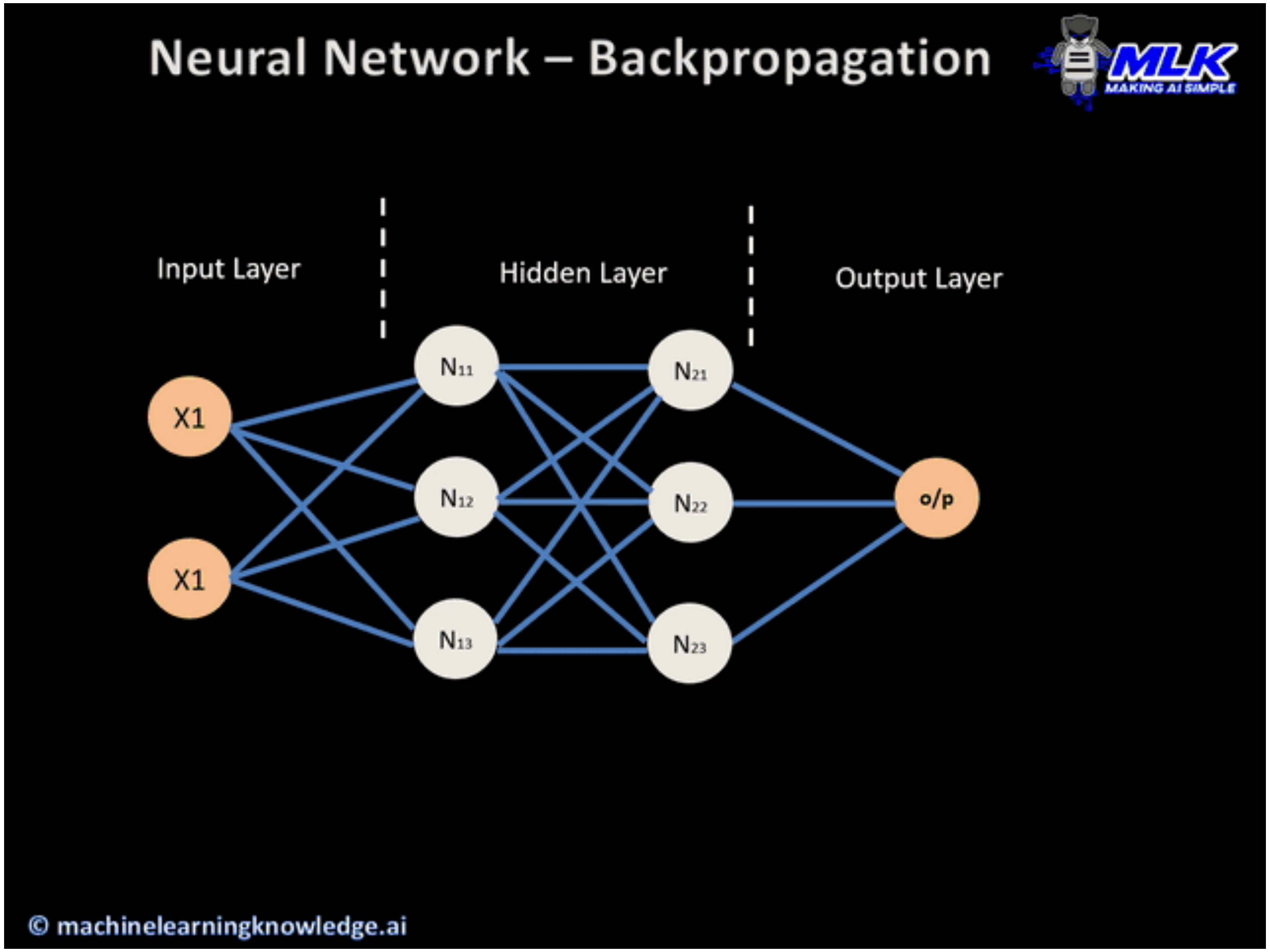
1. input Layer:
   - The input data is fed into the network.
2. Hidden Layers:
   - The input data is transformed by the neurons in the hidden layers using weights, biases, and activation functions.
   - For each neuron, compute: $z = \sum(w \cdot x) + b$

     where w is the weight, x is the input, and b is the bias.
   - Apply an activation function (e.g., ReLU, sigmoid) to z:

     $$a = \text{activation}(z)$$
3. Output Layer:
   - The final layer transforms the activations of the last hidden layer into the output predictions.

Training a Neural Network

Neural networks learn by making mistakes, learning from them, and adjusting their parameters repeatedly to improve and achieve more accurate predictions.



Training a Neural Network

Role of image dimension and batch size

- When training a neural network, multiple samples (e.g., images) are processed together in a single forward and backward pass. We call it a batch of samples.
- The batch size is a hyperparameter that determines the number of samples processed before the model's internal parameters (weights) are updated.
- For example, when you have a batch of 32 images, each with dimensions 20 by 20 pixels, you are feeding the entire batch into the neural network at once. Each image can be thought of as having 400 (20x20) input features.

Training a Neural Network

Example:

Input Representation:

- Each 20x20 image can be flattened into a single vector of 400 features (pixels).
- Thus, each image is represented as a vector of length 400.

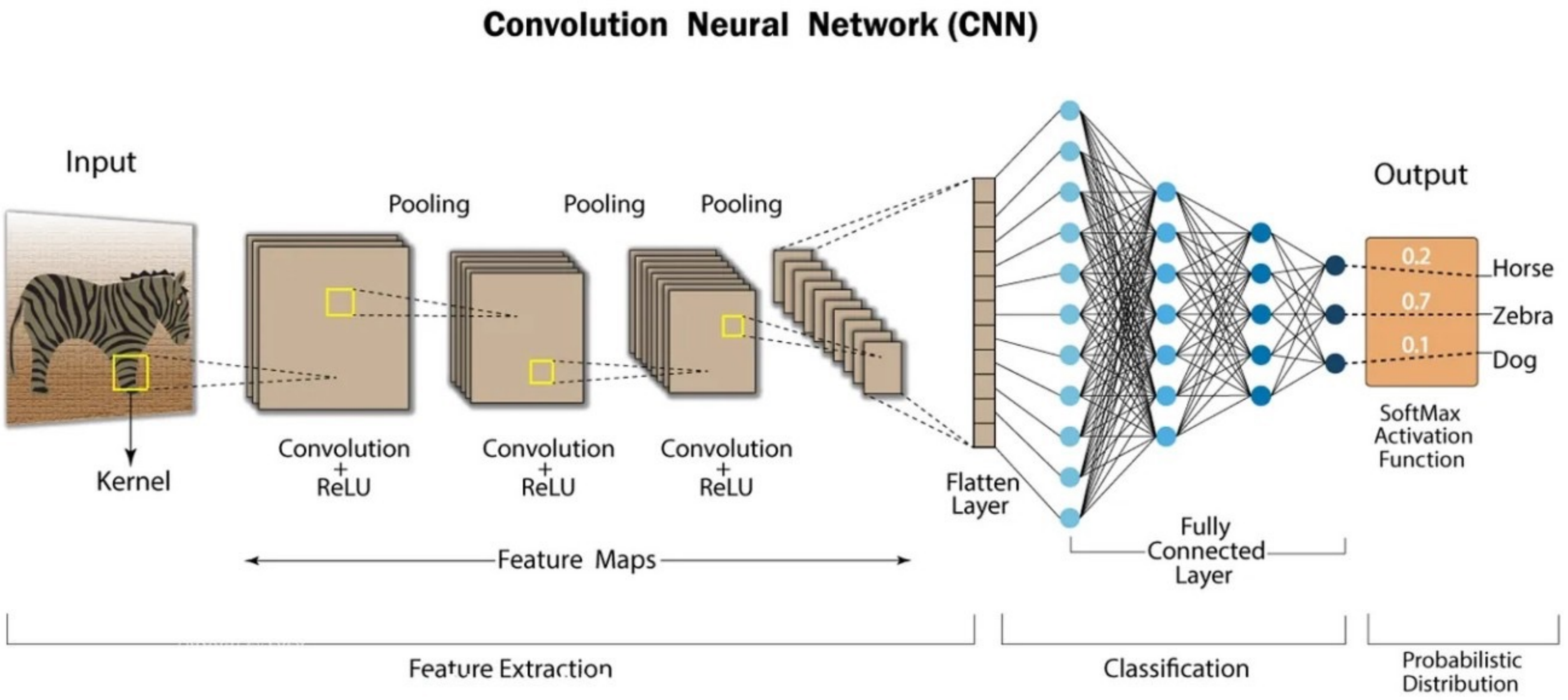
Batch Input Shape:

- A batch of 32 such images would be represented as a matrix where each row is a flattened image.
- The input to the network for one batch would therefore be a matrix of shape (32, 400).

Convolutional Neural Networks (CNNs)

Definition: CNNs are a class of deep learning algorithms designed for processing structured grid data like images.

Purpose: Effective for image recognition, classification, and segmentation tasks.



CNN Architecture

Convolutional Layer: Applies filters (kernels) to the input image to create feature maps.

- The purpose of a convolutional layer is to extract local features from the input data. It does this by applying convolution operations.

- Filters/Kernels:
  - A convolutional layer uses several small filters (or kernels) to scan the input data.
  - Each filter is a small matrix of weights (e.g., 3x3 or 5x5), and it moves across the input image to produce a feature map.
  - The filters are learnable parameters that get updated during the training process.

CNN Architecture

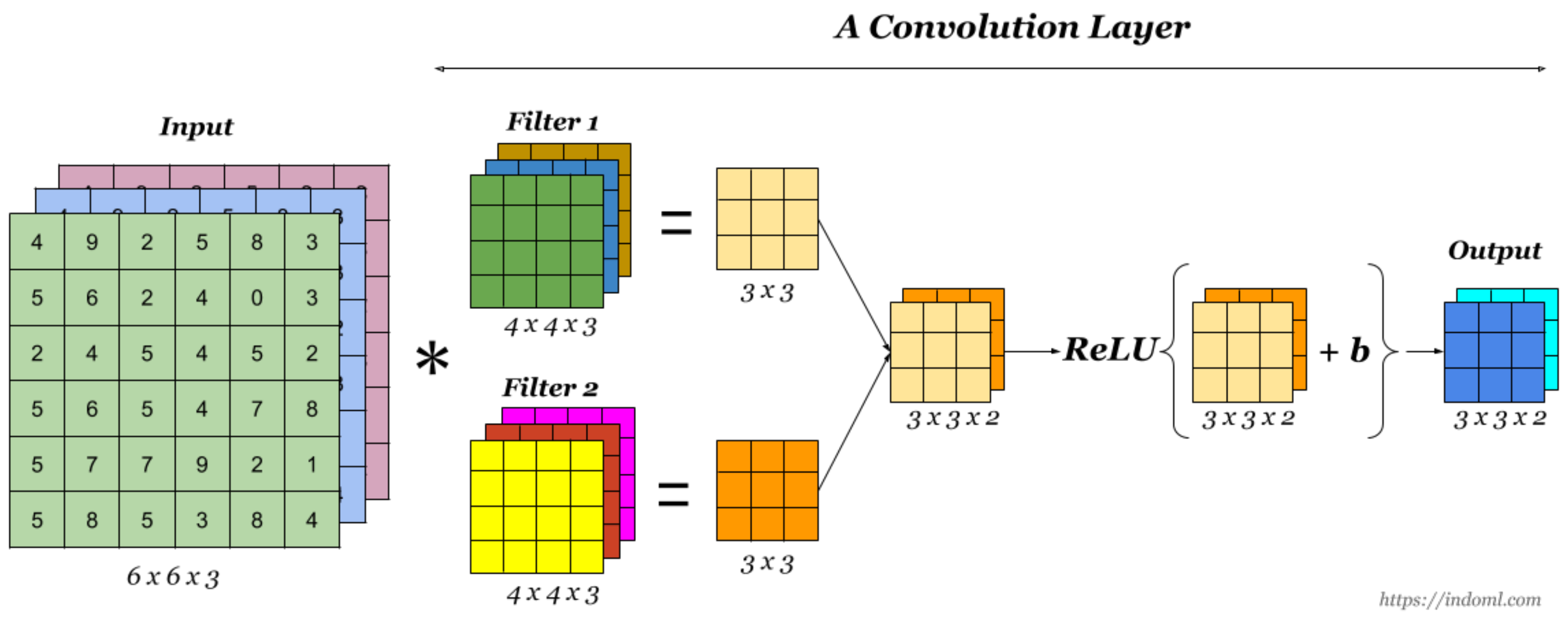


Image credit

CNN Architecture

Convolutional Layer (continue):

– **Stride**: Determines how much the filter moves at each step. A stride of 1 means the filter moves one pixel at a time.

– **Padding**: Adds extra pixels around the border of the input to control the spatial dimensions of the output.

– **Activation Function**: Often an activation function (like ReLU) is applied to the feature maps to introduce non-linearity.

CNN Architecture

Pooling Layer:

Reduces the spatial dimensions of the feature maps.

Max Pooling:

- **Operation**: For each patch of the input, the max pooling layer outputs the maximum value in that patch.
- **Purpose**: It helps to capture the most prominent features (e.g., edges, corners) from the feature maps.
- **Example**: With a 2x2 max pooling operation:

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \quad \text{max pool} \rightarrow 4$$

Recurrent Neural Networks (RNNs)

RNNs are a type of neural network designed to handle sequential data by maintaining a memory of previous inputs.

The term "recurrent" refers to the ability of these networks to retain information about previous inputs and use this information to influence the current output.

- Purpose: Effective for tasks where the order of data is important, such as time series analysis, natural language processing (text), and speech recognition.

Recurrent Neural Networks (RNNs)

Sequential Data:

- Many real-world data types are sequential. Examples include time series data (e.g., stock prices), language data (e.g., sentences), and audio data (e.g., speech).
- The current input in these sequences often depends on previous inputs. For example, the meaning of a word in a sentence can depend on the words that came before it.

RNN Architecture

Components:

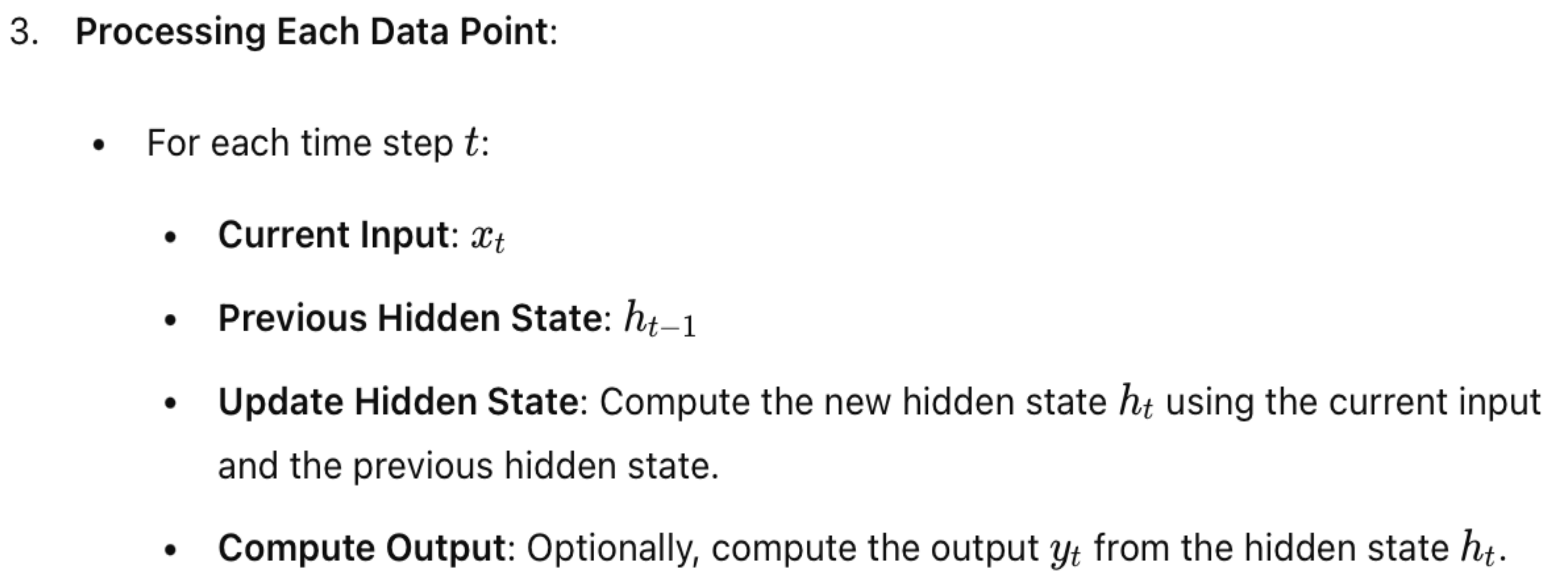
- Input Layer: Receives the sequence of inputs.
- Hidden State: Think of the hidden state as the memory of the network. This memory gets updated every time a new piece of data is processed.
- Output Layer: Produces the final output for each step in the sequence.

How RNNs Work

1. Initialization
   - Hidden State: Initialize the hidden state, usually to a vector of zeros or some learned initial state.

2. Forward Propagation
   - Forward propagation in RNNs involves processing each data point in the sequence one at a time, updating the hidden state at each step. Let's break it down:

How RNNs Work



3. **Processing Each Data Point:**

   - For each time step t:
     - **Current Input**: x_t
     - **Previous Hidden State**: h_{t-1}
     - **Update Hidden State**: Compute the new hidden state h_t using the current input and the previous hidden state.
     - **Compute Output**: Optionally, compute the output y_t from the hidden state h_t.

4 . Calculate Loss:
   Compute the loss over the entire sequence (or batch).

How RNNs Work

Backward Propagation Through Time (BPTT)

1. Compute Gradients:
   - Calculate gradients of the loss with respect to network parameters (weights and biases).
2. Update Parameters:
   - Use an optimization algorithm to update the network parameters.

Repeat for Each Epoch

1. Shuffle Data (optional):
   - Shuffle the dataset to ensure the network doesn't learn the order of the data.
2. Repeat Forward and Backward Propagation:
   - Process all batches and sequences as described above.

Applications of RNNs

- Natural Language Processing: Machine translation, sentiment analysis, text generation.
- Speech Recognition: Converting spoken language into text.
- Time Series Analysis: Predicting stock prices, weather forecasting.
- Others: Music generation, video processing.