Introduction to Deep Learning and Neural Networks

Table of Contents

Learning Objectives

By the end of this unit, you will:

- Understand the need for deep learning.
- Learn applications of deep learning in areas like social media.
- Analyze the structure and working of neural networks.
- Compare biological neurons with artificial neurons.
- Understand the forward propagation process in perceptrons.
- Explore activation functions and their roles.
- Discuss Multi-Layer Perceptrons (MLPs).
- Learn how to improve neural network performance

Course Outline

1. **Deep Learning Overview**
2. **Understanding Neural Networks**
3. **Perceptron and Its Working**
4. **Activation Functions**
5. **Introduction to Multi-Layer Perceptron (MLP)**

Deep Learning

- **Definition**: Deep learning is a subset of machine learning inspired by neural networks in the human brain.
- **Advantages**:

    - Handles high-dimensional data effectively (solves the curse of dimensionality).
    - Learns features automatically from large datasets.
- **Key Applications**:
    - Google Images for classification.
    - Siri for speech recognition.
    - Netflix for recommendations.
    - AlphaGo for game strategy learning.
- **Enablers of Deep Learning**:
    - **Big Data**: Availability of large datasets.
    - **Hardware**: Parallelized GPUs for efficient computation.
    - **Software**: Libraries like TensorFlow and PyTorch.
    - **Community**: Open-source research and collaboration.

Understanding Neural Networks

- **Biological vs. Artificial Neurons**:
    - Biological neurons transfer information via dendrites, synapses, and axons.
    - Artificial neurons mimic this with inputs, weights, activation functions, and outputs.

Perceptron

- **Definition**: A perceptron is the building block of deep learning and forms the simplest artificial neural network.
- **Working**:
    - Computes a **weighted sum** of inputs.
    - Adds a **bias** to the sum.
    - Applies a **non-linear activation function**.
    - Performs **forward propagation** to produce an output.

Activation Functions

- **Purpose**: Introduce non-linearity to model complex patterns.
- **Types**:
    - **Sigmoid**: Outputs values between 0 and 1. Used for probabilities but suffers from vanishing gradients.
    - **Tanh**: Outputs values between -1 and 1. Handles vanishing gradients better than Sigmoid.
    - **ReLU**: Most common; outputs values between 0 and infinity. Efficient but can cause "dead neurons."
    - **Softmax**: Used in the final layer for multi-class classification.

Multi-Layer Perceptron (MLP)

- **Structure**: Consists of:
    - Input Layer
    - Hidden Layers (introduce non-linearity)
    - Output Layer
- **Training Process**:
    - Calculate the **cost function**: Difference between expected and actual outputs.
    - Use **backpropagation** to adjust weights.
    - Repeat until the cost is minimized.

Improving Neural Network Performance

- Techniques:
    - Regularization (e.g., L1, L2).
    - Optimizers like SGD, Adam.
    - Use of advanced architectures (e.g., CNNs, RNNs).
    - Hyperparameter tuning for learning rates, batch sizes, etc.

Example with Python: Digit Classification

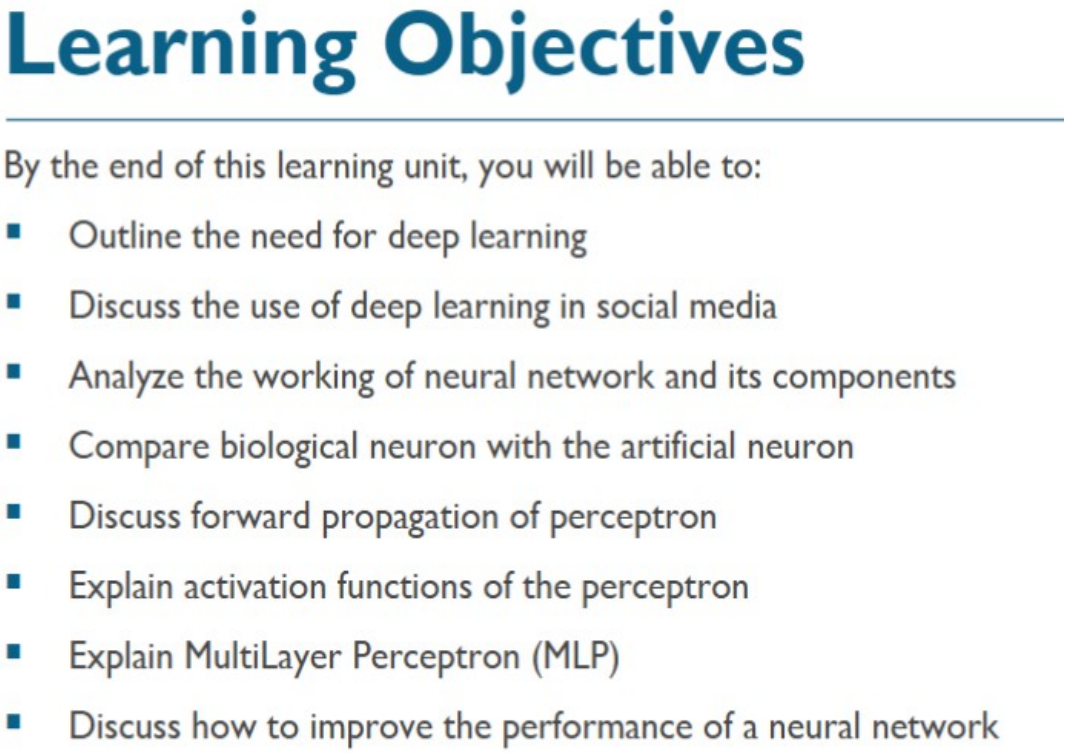
Below is a Python example using scikit-learn to classify digits with a neural network.

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# Load dataset
digits = load_digits()
X, y = digits.data, digits.target

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Define the model
mlp = MLPClassifier(hidden_layer_sizes=(64, 64), activation='relu', max_iter=500,
random_state=42)

# Train the model
mlp.fit(X_train, y_train)

# Test predictions
y_pred = mlp.predict(X_test)

# Evaluate performance
accuracy = accuracy_score(y_test, y_pred)
```

```
print(f"Accuracy: {accuracy * 100:.2f}%")
```

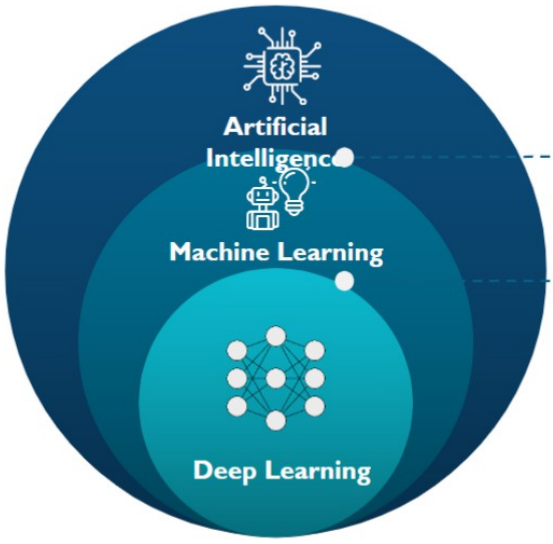
Conclusion

Deep learning is revolutionizing industries by enabling complex, large-scale data analysis. The foundation is neural networks, which use perceptrons and activation functions for efficient learning. Libraries like TensorFlow and PyTorch, along with growing data availability, make deep learning accessible for diverse applications.



Deep Learning: A Part of Machine Learning

Deep Learning is a subset of Machine Learning inspired by neural networks in the human brain.

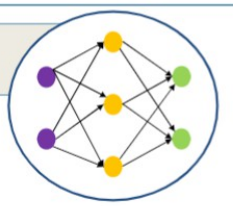


Ability of machines to learn without being programmed.

Learning the underlying features in data using algorithms called **Artificial Neural Networks (ANNs)** which are inspired by the structure and function of the human brain.

What is Deep Learning?

ANNs are having many layers.



- They are the core of deep learning.
- They are powerful, versatile and scalable.
- They find applications in:

Google Images

Classifying billions of images

Apple's Siri

Powerful Speech Recognition

Netflix

Recommending movies to watch for millions of users

DeepMind's AlphaGo

Learning to beat the world champion at the game of Go

AlphaGo

Why Deep Learning?

Ideas of neural networks have been around since 1940s, so what has led to the exploding success and demand for these techniques today?

Big Data
Large data sets, digitisation of data makes collection and storage simpler.

Hardware
GPUs, parallelised processing enables efficient large-scale execution of algorithms.

Software
Tools such as TensorFlow and PyTorch streamline the process of implementing these algorithms.

Community
GitHub and other such communities let the researchers share the developments.

Deep Learning: Improvement Over ML

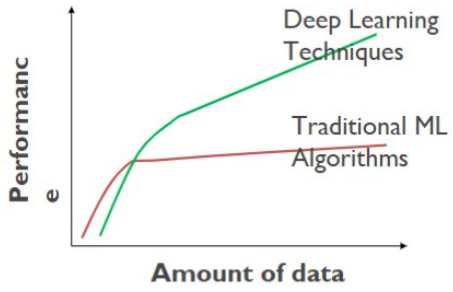
Deep learning comes to the rescue to overcome the curse of dimensionality.

Curse of dimensionality

Traditional ML algorithms are not effective when working with high dimensional data (100 or 1000 or even more features) as feature extraction becomes complicated.

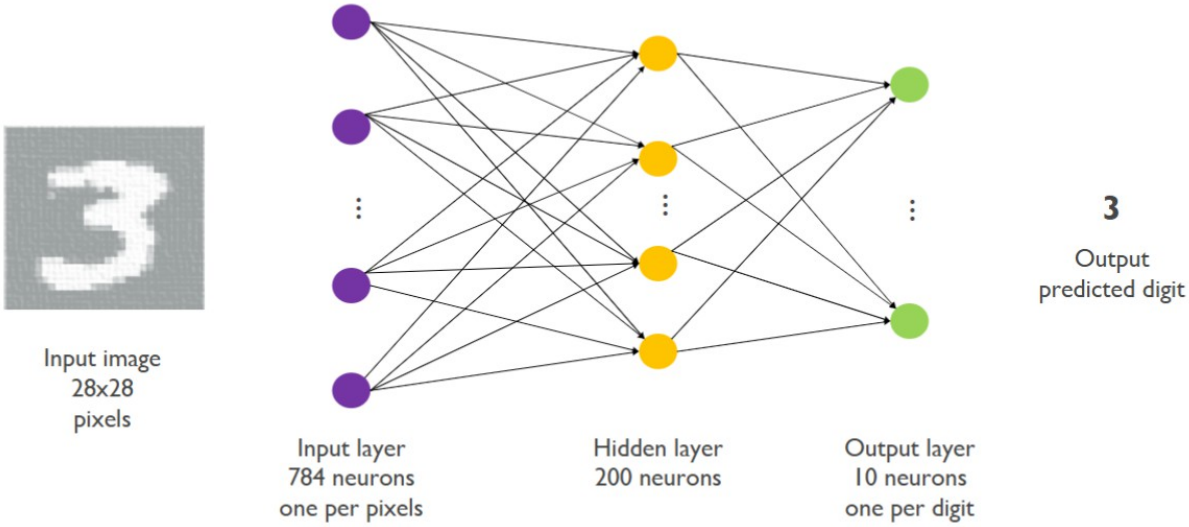
Resolved by deep learning

Deep learning models are capable of learning to focus on the right features on their own.



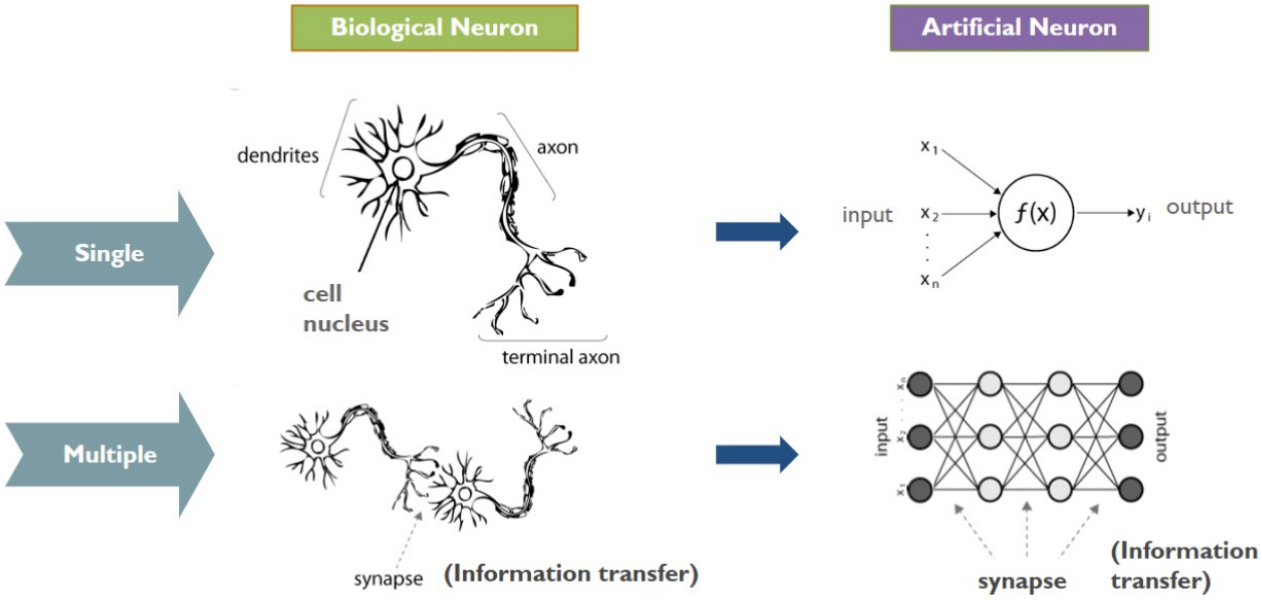
Deep learning and Machine learning comparison

How Does a Machine Identify a Digit?



Note: This is a general representation of each layer. The actual number of neurons in each layers and the number of layers depend upon the input data values and the complexity of the characteristics of input data.

Biological vs. Artificial Neuron



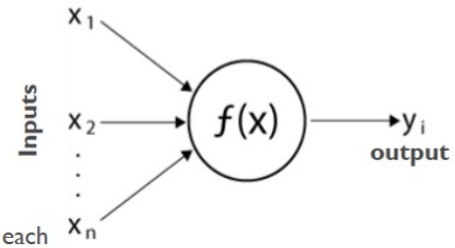
Human Brain vs. Neural Network: Terminology

Neuron	Artificial Neuron
Cell Nucleus	Node
Dendrites	Input: x_1, x_2
Synapse	Weights
Axon	Activation function
Terminal Axon	Output: y_1

What is Perceptron?

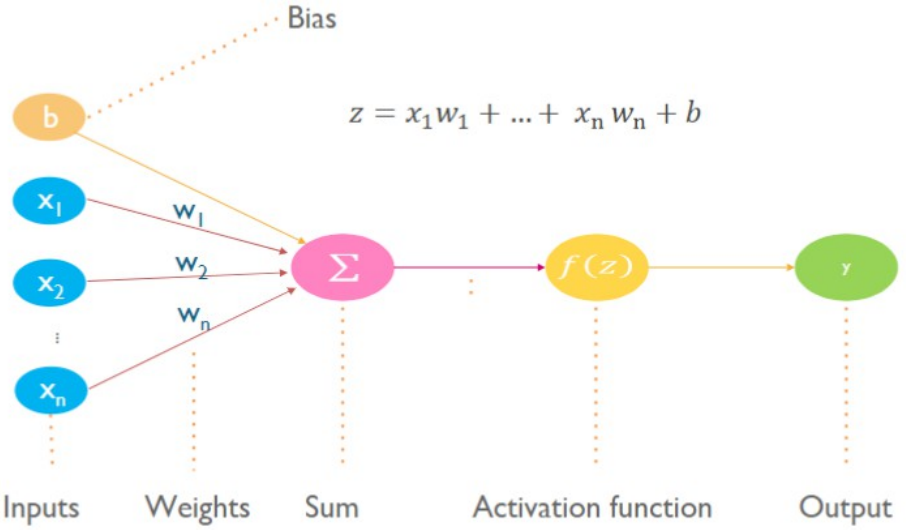
- It is the structural building block of deep learning.
- It has the simplest Artificial Neural Network (ANN) architecture.

Working of a perceptron:



- The perceptron has a set of inputs and outputs and a weight associated with each input.
- It computes a weighted sum of inputs.
- It adds a bias to the sum.
- It implements a nonlinear function (activation function) to produce the outputs.
- This process is called **forward propagation**.

Perceptron: Forward Propagation



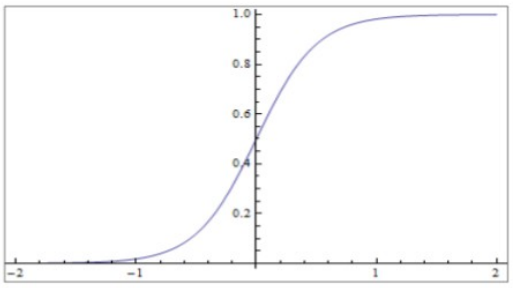
$$y = f\left(b + \sum_{i=1}^{n} x_i w_i\right)$$

Introduction to Activation Function

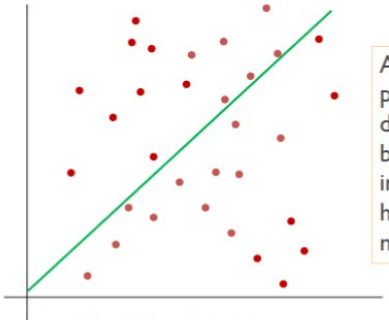
- Activation function, also known as a transfer function, introduces nonlinearity in neural networks.
- This nonlinear transformation is introduced to learn the complex underlying patterns in the data.
- Without this function, a neuron simply resembles a linear regression.
- For example: Sigmoid function

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

$$z = b + \sum_{i=1}^{n} x_i w_i$$

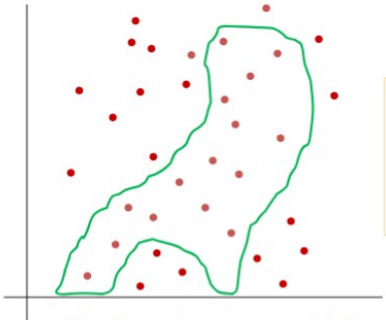


Given below are the classification scenarios:



A linear function produces linear decision boundaries irrespective of how deep the network is.

Classifying data into two labels using a linear function

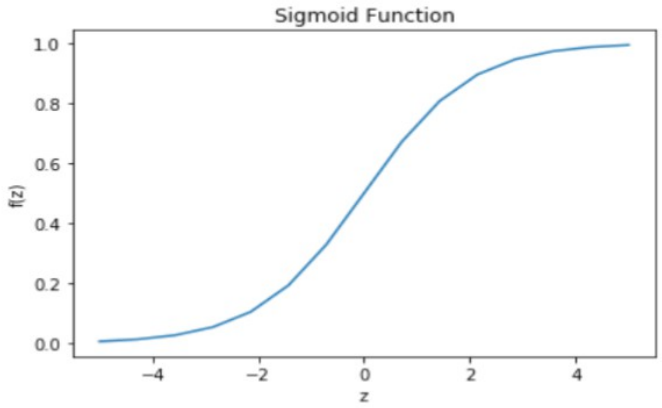


Nonlinearities allow us to model complex functions with improved capabilities.

Classifying data into two labels using a nonlinear function

Sigmoid Function

- It is one of the commonly used functions.
- It scales all values between 0 and 1, thus used for predicting probabilities.

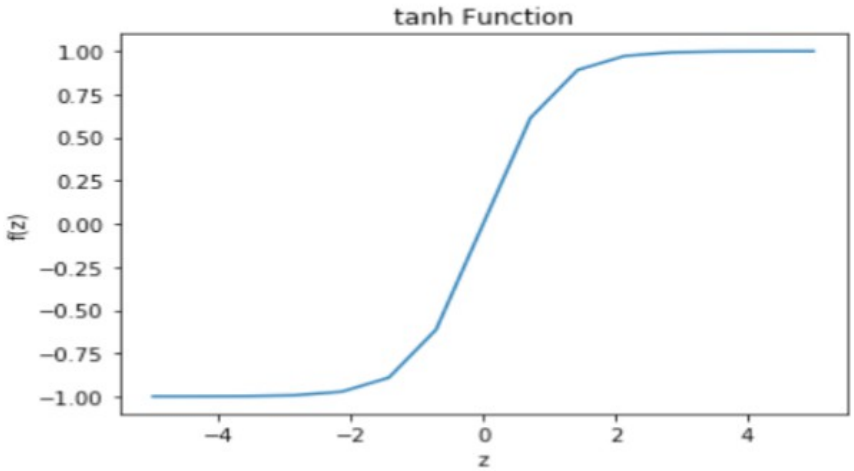


$$f(z) = \frac{1}{(1 + e^{-z})}$$

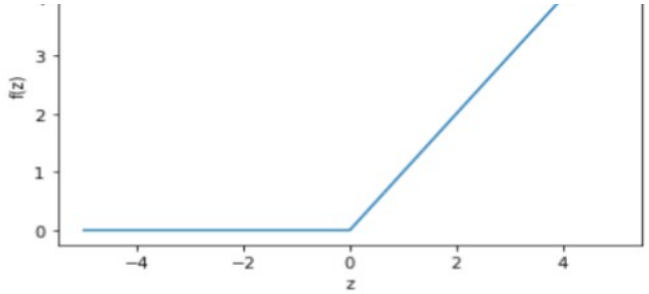
Problem of "vanishing gradients" arises, which makes it unsuitable for backpropagation.

Tanh Function

- A hyperbolic tangent (Tanh) function maps input values to outputs between −1 and 1.
- This function solves the "vanishing gradients" problem to some extent.



$$f(z) = \frac{(1 - e^{-2z})}{(1 + e^{-2z})}$$



f(z) = { 0, for z<0 ; z, for z≥0

Softmax Function

- It is a generalization of Sigmoid function.
- Softmax maps output to values between 0 and 1 such that their sum is 1.
- Thus, Softmax is applied to the final layer of a network in a multiclass classification.

z_n	$f(z_n)$
-5	0
-3	0.0003
0	0.0064
2	0.0471
5	0.9462

$$f(z_n) = \frac{e^{z_n}}{\sum_{i=1}^{n} e^{z_n}}$$, where n is the number of classes

Comparing Activation Functions

Activation function	Benefits	Drawbacks	Applications
Sigmoid	Small changes in input resulting in large changes in output	Saturation near 0 and 1, resulting in vanishing gradient; also, outputs not being zero-centered	Acts as a good classifier
Tanh	Outputs zero centered, hence preferred over Sigmoid.	Wider range as compared to Sigmoid which results in faster learning	Gives best performance in Convolutional Neural Network (CNN) applications
ReLU	Nonsaturating and easier to implement expensive operations	ReLU units being fragile and having possibility to "die" during training	Gives best performance in Classification using MultiLayer Perceptron (MLP), Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) applications
Softmax	Helps in probabilistic interpretation	Similar to Sigmoid	Useful in the final layer of a multi-class classification
Tanh	Outputs zero centered, hence preferred over Sigmoid.	Wider range as compared to Sigmoid which results in faster learning	Gives best performance in Convolutional Neural Network (CNN) applications
ReLU	Nonsaturating and easier to implement expensive operations	ReLU units being fragile and having possibility to "die" during training	Gives best performance in Classification using MultiLayer Perceptron (MLP), Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) applications
Softmax	Helps in probabilistic interpretation	Similar to Sigmoid	Useful in the final layer of a multi-class classification

Improving the Performance of a Neural Network