Introduction To Deep Learning & Neural Networks with keras (M2)

Created	@December 8, 2025 7:23 PM
Module Code	IBMM02: Introduction to Deep Learning

Module 1: Introduction to Deep Learning

Overview of Deep learning

- Deep learning is a core area in data science, driving recent breakthroughs in various applications.
- Neural networks are the foundation of deep learning applications.

Notable Applications

1. **Color Restoration**
    - Grayscale images are automatically converted to color.
    - Uses **Convolutional Neural Networks (CNNs)**.
2. **Speech Enactment**
    - Audio clips are synthesized with videos so that lip movements match speech.
    - Can extract audio from one video and sync it to another.
    - Example: Barack Obama video lip-syncing system.
    - Uses **Recurrent Neural Networks (RNNs)**.
3. **Automatic Handwriting Generation**

- Converts typed text into realistic cursive handwriting in various styles.
- Developed by Alex Graves at the University of Toronto.
- Uses **RNNs**.

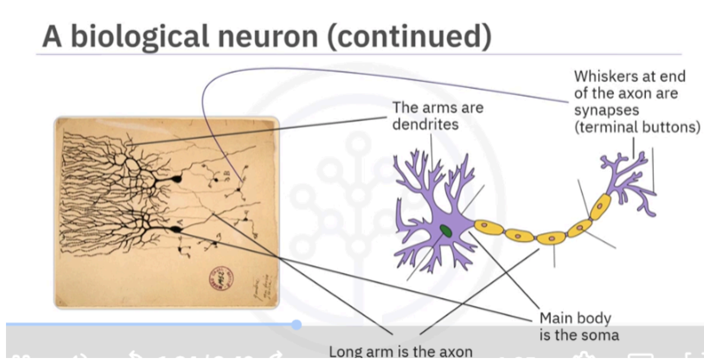
4. **Other Applications**
   - **Automatic machine translation** – translating text in images.
   - **Adding sounds to silent movies** – deep learning selects appropriate sounds.
   - **Image/object classification**.
   - **Self-driving cars**.
   - **Chatbots**.
   - **Text-to-image generators**.

Neurons and Neural Networks

Deep learning is inspired by the structure and behavior of the human brain. To understand artificial neural networks, we first look at **biological neurons**.

🧠 1. Biological Neuron: The Main Parts

A biological neuron has 4 key components:



1. Soma (Cell Body)

- The “main body” of the neuron.
- Contains the **nucleus**, which processes incoming signals.

2. Dendrites

- Branch-like structures.
- Receive incoming electrical signals (“information”) from other neurons.

3. Axon

- A long tube-like structure that carries the processed signal away from the soma.

4. Synapses (Terminal Buttons)

- Located at the end of the axon.
- These connect to other neurons and transmit the output signal.

📌 **Flow of information in a neuron:**

Dendrites → Soma → Axon → Synapse → Next neuron

🔗 2. How Biological Neural Networks Work

Each neuron may connect to **thousands** of other neurons.

Learning in the brain works by strengthening certain connections.

- When some neural pathways fire repeatedly → they become stronger.
- Stronger connections = more likely to activate in the future.
- This is how the brain “learns”.

🤖 3. Artificial Neurons (Artificial Neural Networks)

Artificial neurons are *simplified mathematical versions* of biological neurons.

They mimic the same overall structure:

Artificial Dendrites → Inputs

These are the features (x_1, x_2, x_3...).

Soma → Weighted Sum

The neuron combines inputs using weights:

$z=w1\times1+w2\times2+...+b$

Nucleus → Activation Function

This adds non-linearity (e.g., ReLU, sigmoid).

Axon → Output

The output becomes the input to other neurons.

Learning → Adjusting Weights

Just like biological neurons strengthen connections, artificial neural networks learn by adjusting weights during training (via backpropagation).

🧩 4. Summary (Super Short)

- Biological neurons have dendrites, soma, axon, and synapses.
- Signals flow from dendrites → soma → axon → synapse.
- Learning happens by strengthening frequently used connections.
- Artificial neurons copy this behavior using:
    - inputs
    - weights
    - sums
    - activation functions
    - outputs
- Deep learning = many artificial neurons working together.

Artificial Neural Networks

1. What is an Artificial Neuron?

Think of an artificial neuron (also called a **perceptron**) as a tiny calculator.

It:

1. Takes input values (like x_1, x_2, ...)
2. Multiplies them by weights (w_1, w_2, ...)
3. Adds a bias
4. Passes the result through an activation function
5. Gives an output

Mathematically:

$$z = w_1x_1 + w_2x_2 + \cdots + b$$

$$a = \sigma(z)$$

Where:

- **z** = weighted sum
- **a** = final output
- **σ()** = activation function (e.g., sigmoid)

---

2. Why do we need Activation Functions?

Without activation functions, a neural network is just a **big linear regression model**.

Activation functions (like **sigmoid, ReLU, tanh**) introduce **non-linearity**, helping networks learn complex tasks like:

- image classification
- speech recognition
- language translation

Example (Sigmoid):

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

---

3. Layers in a Neural Network

A neural network has:

Input Layer

- Receives features (like pixel values, temperature, etc.)

Hidden Layers

- Do all the internal calculations
- Can have many neurons and many layers

Output Layer

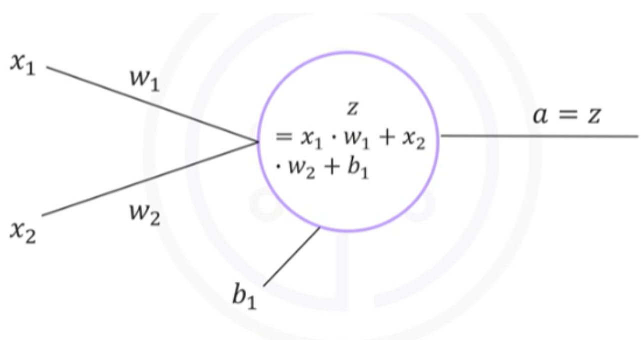
- Produces predictions

  (e.g., 0/1 for classification, or a numeric value for regression)

---

4. Forward Propagation (very important)

This is the process of passing data from **input → hidden layers → output**.



Example:

Let:

- Input: **x = 0.1**

- Weight: **w = 0.15**
- Bias: **b = 0.4**

Step 1: Compute weighted sum

$$z = wx + b = (0.15)(0.1) + 0.4 = 0.415$$

Step 2: Apply activation (sigmoid)

$$a = \sigma(0.415) = 0.6023$$

This a becomes input for the next neuron if there is another layer.

This **a** becomes input for the next neuron if there is another layer.

🚀 Easy Real-Life Analogy

Think of a neural network like a decision-making process:

Input

You see a fruit: color, shape, size.

Hidden Layers

Your brain processes these attributes:

- "Is it round?"
- "Is it orange?"
- "Small or big?"

Output

You decide: **"It's an orange!"**

Learning = **finding the best numbers (called parameters)** for your model.

In linear regression, the parameter is:

- **w** (weight)
- sometimes also **b** (bias)

In neural networks, parameters are:

- **many weights**
- **many biases**

Learning = adjusting these weights and biases until the model makes **minimum error**.

📝 Simple Summary

- A neural network is made of **layers of artificial neurons**.
- Each neuron does:

  weighted sum → add bias → activation function → output
- Forward propagation is the step-by-step movement of data from input to output.
- Activation functions allow the network to learn **complex patterns**.