Since 2011

RIG

Realistic Infotech Group

Our outcomes are over 5000 trainees.

Get IT Right from RIG

Artificial Intelligence Engineering (Level-1)

Learn, Create, and Shine

Digital Space

Content

- Module 1: Introduction to AI and Machine Learning
- Module 2: Linear Algebra, Statistics and Probability for AI
- Module 3: Neural Network Architecture
- Module 4: Building Machine Learning Models
- Module 5: Deep Learning Concepts
- Module 6: Python Data Structure
- Module 7: Data Handling with Pandas and NumPy
- Module 8: Python for AI
- Module 9: Classification AI Project
- Module 10: Prediction AI Project

Artificial Intelligence Engineering (Level-1)

Module 3: Neural Network Architecture

Content

- Introduction to Neural Networks
- History of Neural Network
- Applications of Neural Networks
- Concepts of Neural Networks
- Types of Neural Networks
- Neural Network Architecture
- Evaluating Model Performance

Leaning Outcomes

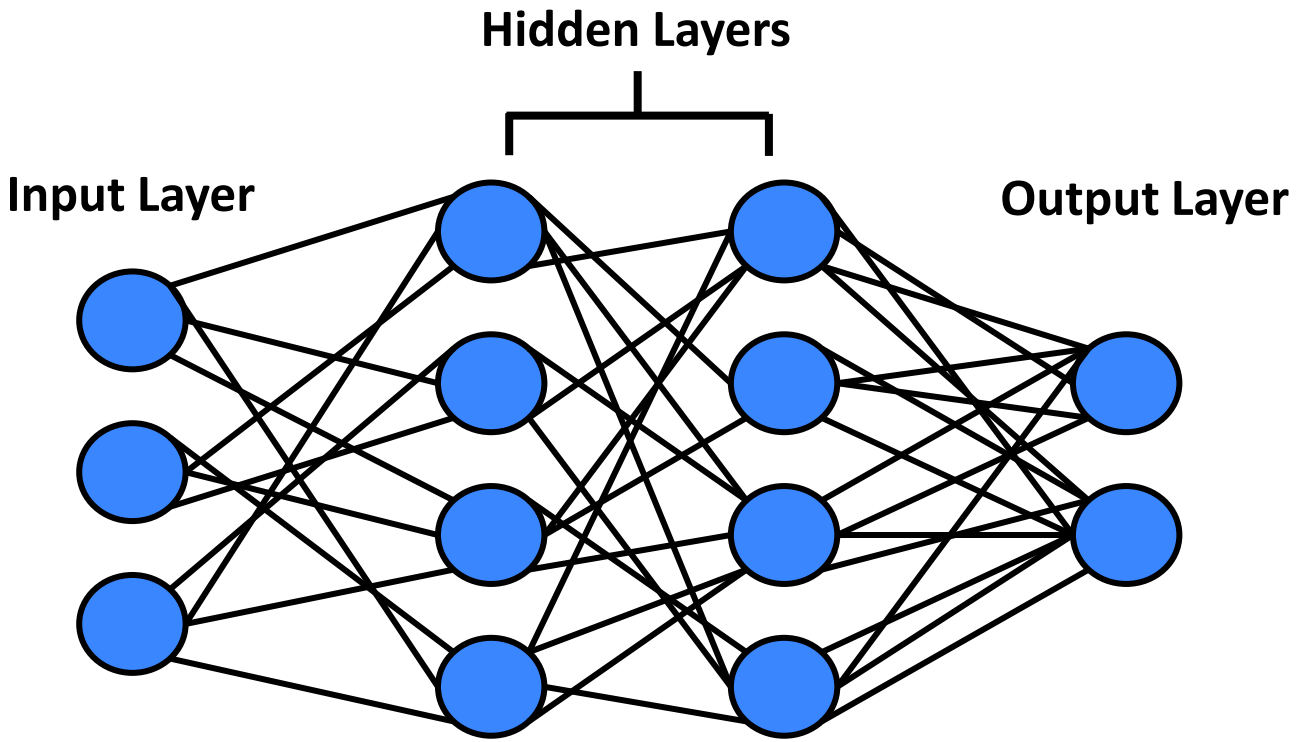
- Neural networks mimic the human brain to solve AI and machine learning tasks by processing data through interconnected neurons.
- They play a vital role in modern technology, evolving from foundational models like the Perceptron to advanced deep learning architectures.
- Historical milestones, such as backpropagation, and contributions from pioneers like McCulloch and Pitts, laid the groundwork.
- Neural networks excel in applications like image recognition, speech processing, and decision-making across healthcare, finance, and robotics.
- Key concepts include neurons, weights, layers, and learning processes like forward and backpropagation.

Leaning Outcomes

- Various types, such as CNNs and RNNs, cater to specific use cases.
- Architectures consist of input, hidden, and output layers, with hyperparameters optimizing performance.
- Model evaluation relies on metrics like accuracy and F1-score, and tools like ROC curves, ensuring reliability.
- Emerging hybrid models and trends continue to expand the potential of neural networks in innovative fields.

Introduction to Neural Networks

Neural networks have become a foundational technology behind many modern AI applications. This section is meant to provide beginners with a clear understanding of what neural networks are, their historical development, and their real-life applications.



Definition of Neural Networks

What is Neural Networks?

- A **Neural Network** is a type of **machine learning model** inspired by the way the human brain processes information. It consists of interconnected units called **neurons**, which mimic the function of biological neurons.

- Neural networks are designed to **recognize patterns**, make decisions, and **learn from data**.

Definition of Neural Networks

Key Concepts

- **Artificial Neurons**: Simplified models of biological neurons.
- **Learning from Data**: Neural networks adjust their internal parameters based on examples to improve performance.
- **Non-linear Problem Solving**: Neural networks excel at solving complex problems that involve patterns, non-linear relationships, and vast amounts of data.

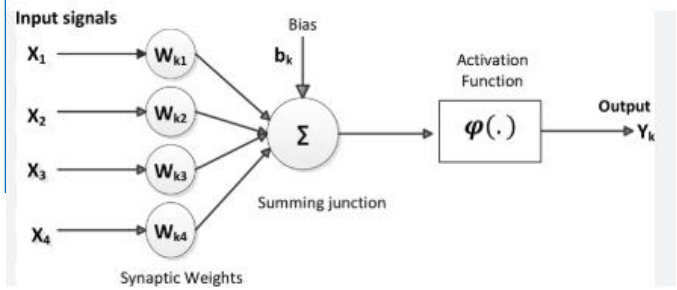
Analogy

- Imagine neural networks as a group of problem-solving experts. Each expert (neuron) contributes a small part of the solution based on its knowledge, and together they arrive at a final answer.

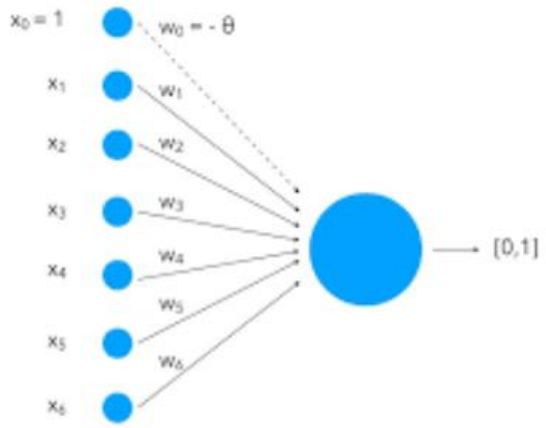
History of Neural Netwc

1943 - The Birth of the Idea

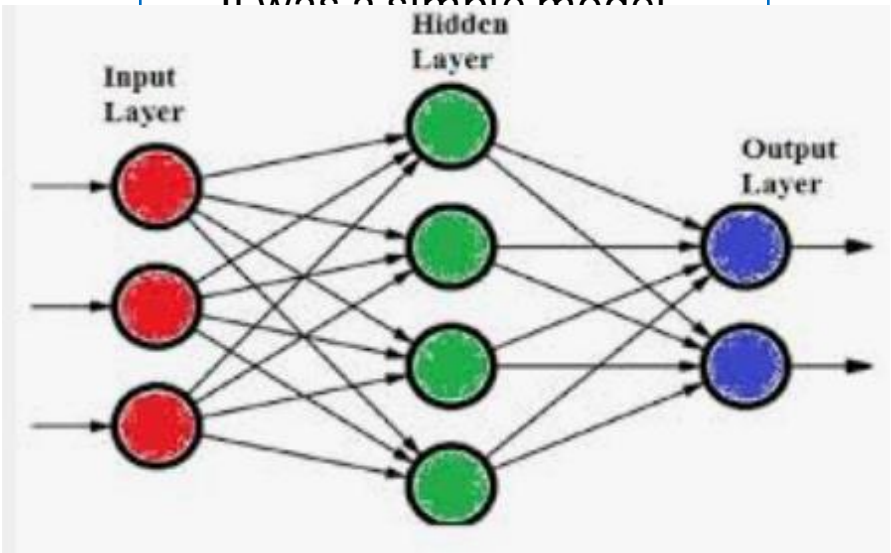
- The concept of artificial neurons was first introduced by **Warren McCulloch** (a neurophysiologist) and **Walter Pitts** (a logician) in their paper, proposing a mathematical model of biological neural networks.



19



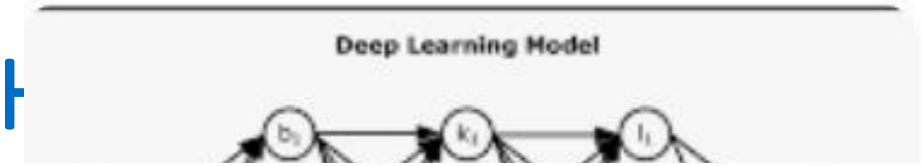
- I
  Perceptron, one of the earliest neural networks. It was a simple model

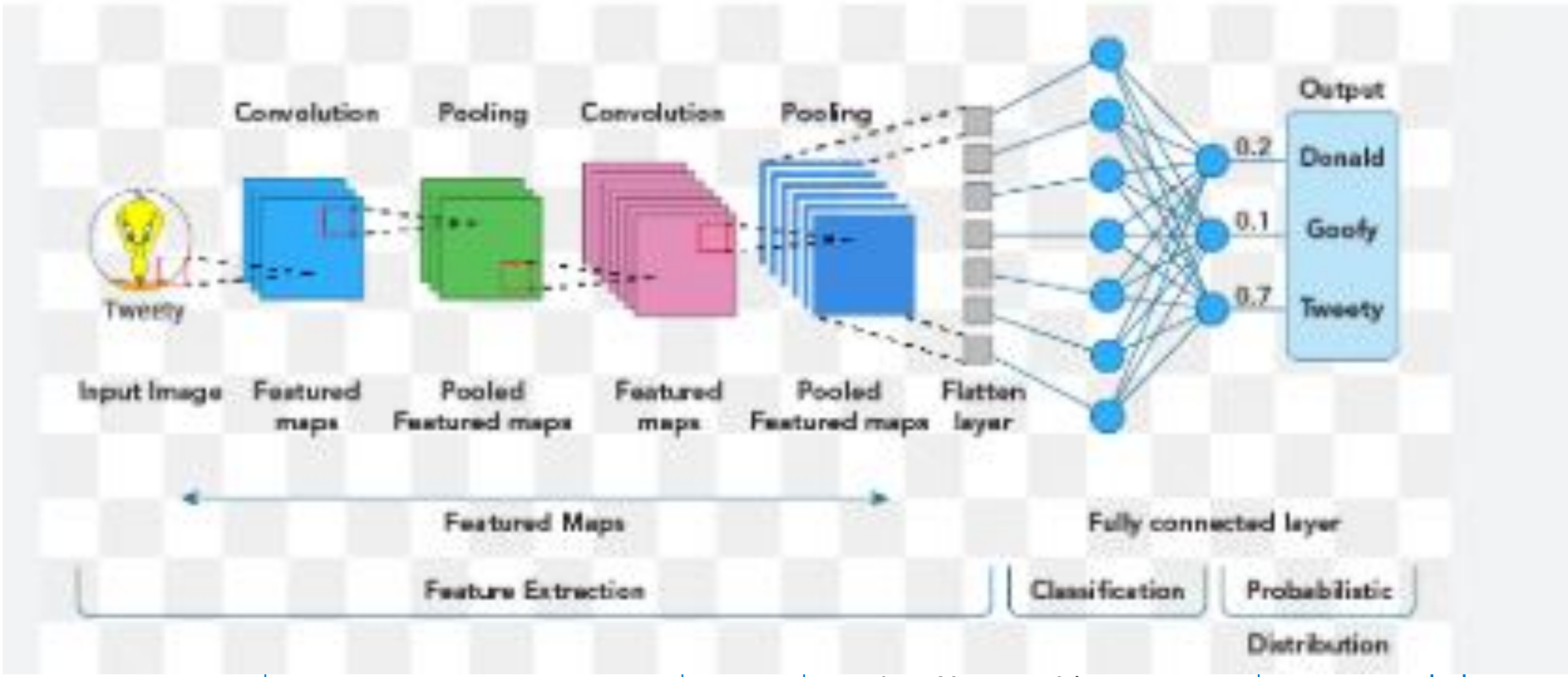


1980s - The Revival with Backpropagation

- **Geoffrey Hinton**, **David Rumelhart**, and **Ronald Williams** popularized the **backpropagation algorithm**, which allowed deeper networks to learn efficiently.
- This breakthrough enabled neural networks to solve more complex problems by using multiple hidden layers.

H vork





what AI can achieve.

Applications of Neural Networks

Neural networks are widely used across various industries, revolutionizing the way we interact with technology.

Applications of Neural Networks

Image Recognition

- **Facial Recognition Systems**: Used by social media platforms and security systems to identify individuals.
- **Medical Imaging**: Assists radiologists in diagnosing diseases like cancer, pneumonia, and COVID-19 from X-rays, MRIs, and CT scans.
- **Object Detection**: Self-driving cars use neural networks to recognize pedestrians, traffic signs, and obstacles on the road.

Natural Language Processing (NLP)

- **Speech Recognition**: Virtual assistants like **Siri**, **Google Assistant**, and Alexa use neural networks to understand spoken language.
- **Text Generation and Summarization**: Tools like **ChatGPT** and **BERT** models are used for generating human-like text, summarizing articles, and automating content creation.
- **Language Translation**: Services like Google Translate rely on neural networks to translate text between languages in real-time.

Self-Driving Cars

- Autonomous vehicles use a combination of **CNNs for visual perception** and **RNNs for decision-making** to navigate roads safely.
- Neural networks help cars detect lane markings, traffic signs, pedestrians, and other vehicles, enabling real-time decision-making.

Applications of Neural Networks

Financial Industry

- **Fraud Detection**: Banks use neural networks to detect fraudulent transactions by recognizing unusual patterns in transaction data.
- **Stock Market Predictions**: Predictive models powered by neural networks analyze historical data and current trends to make stock predictions.

Healthcare

- **Personalized Medicine**: Analyzing patient data to recommend personalized treatment plans.
- **Drug Discovery**: Accelerating the process of discovering new drugs by predicting how different compounds interact with biological systems.

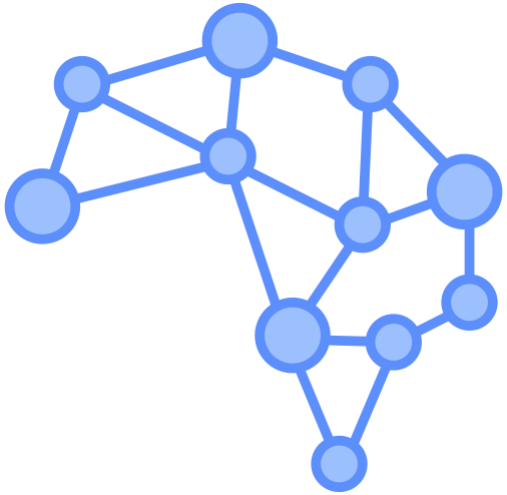
Entertainment and Media

- **Recommendation Systems**: Platforms like **Netflix**, **Spotify**, and **YouTube** use neural networks to recommend content based on user preferences and behavior.
- **Content Creation**: Tools for generating art, music, and even AI-generated videos.

Applications of Neural Networks

Summary

- Neural networks are the backbone of modern AI, enabling machines to learn, adapt, and solve complex tasks.
- The development of neural networks has been a journey from simple models to deep learning architectures that power the technology around us today.
- From image recognition to self-driving cars, neural networks have become an integral part of our daily lives.



Concepts of Neural Networks

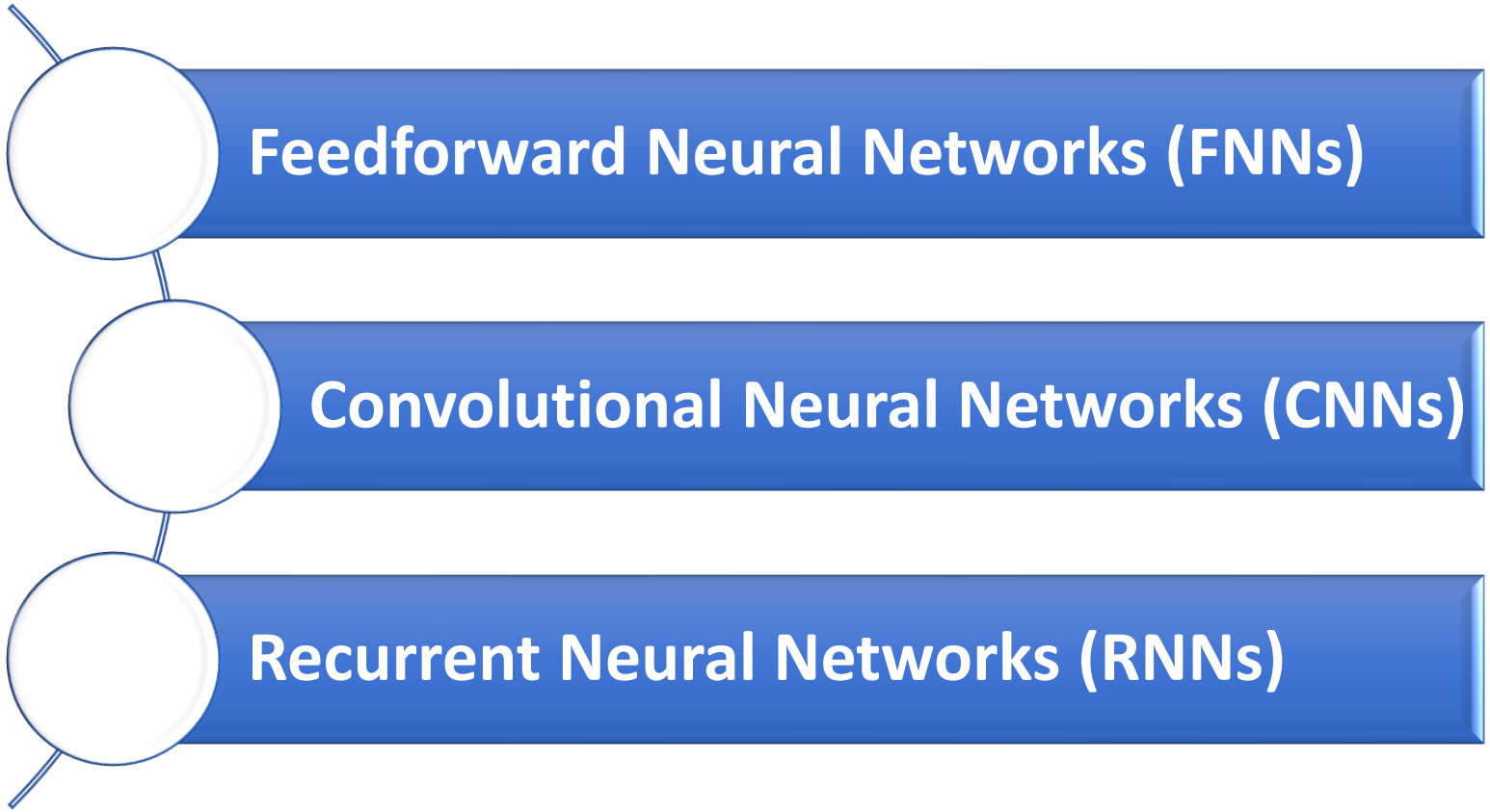
This section provides a deeper understanding of the structure of neural networks, including their layers and types. By the end of this section, you'll have a solid grasp of the basic components and different types of neural networks.

Layer Structure:

Layer	Role	Example Use Case
Input Layer	Receives raw input data	Image pixels, text tokens, numerical data
Hidden Layers	Processes and transforms data	Feature extraction, pattern recognition
Output Layer	Produces final predictions	Class probabilities, regression values

Types of Neural Networks

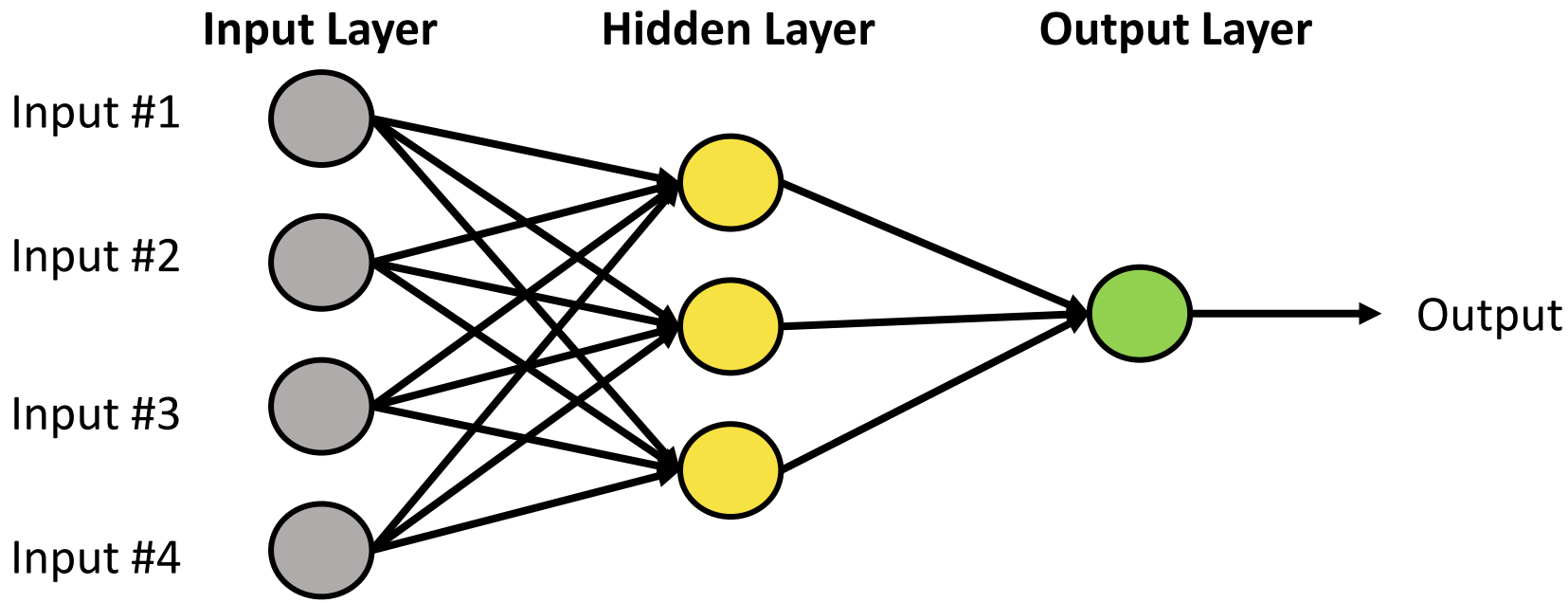
There are several types of neural networks, each designed to handle specific types of data and tasks. Let's explore the most common ones:



Feedforward Neural Network

Feedforward Neural Networks (FNNs)

Definition: The simplest type of neural network where data flows in one direction—from the input layer, through hidden layers, to the output layer.



an example of a feedforward neural networks with one hidden layer (with 3 neurons).

FNNs

Characteristics

- No cycles or loops; the network does not revisit any layer.
- Typically used for problems like **classification** and **regression**.

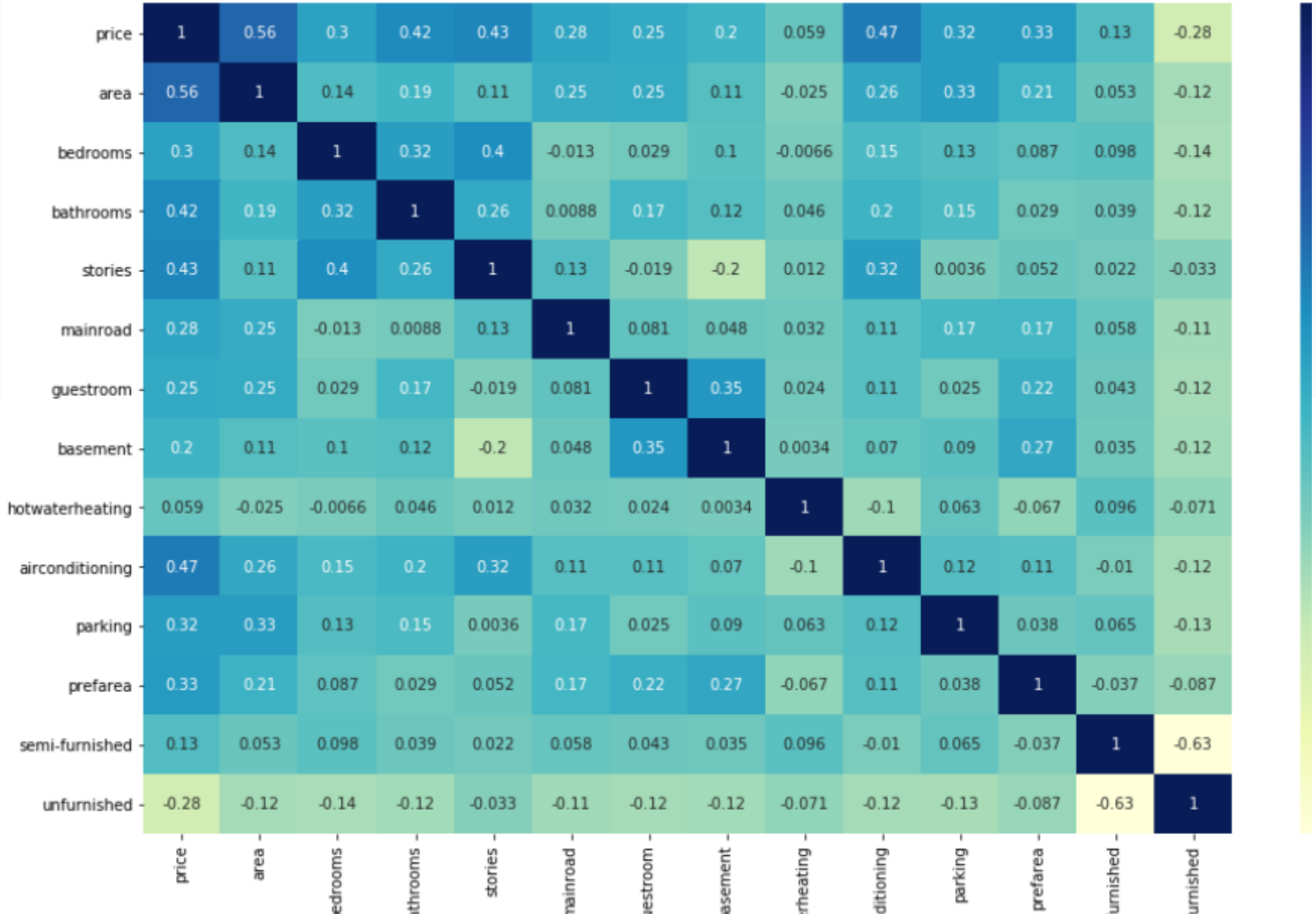
Applications

- Predicting house prices based on features like size, location, etc.
- Email spam filtering.

Example Architecture

- Input Layer → Hidden Layer(s) → Output Layer

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking
0	13300000	7420	4	2	3	yes	no	no	no	yes	2
1	12250000	8960	4	4	4	yes	no	no	no	yes	3
2	12250000	9960	3	2	2	yes	no	yes	no	no	2
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2


price 1 0.56 0.3 0.42 0.43 0.28 0.25 0.2 0.059 0.47 0.32 0.33 0.13 -0.28
area 0.56 1 0.14 0.19 0.11 0.25 0.25 0.11 -0.025 0.26 0.33 0.21 0.053 -0.12
bedrooms 0.3 0.14 1 0.32 0.4 -0.013 0.029 0.1 -0.0066 0.15 0.13 0.087 0.098 -0.14
bathrooms 0.42 0.19 0.32 1 0.26 0.0088 0.17 0.12 0.046 0.2 0.15 0.029 0.039 -0.12
stories 0.43 0.11 0.4 0.26 1 0.13 -0.019 -0.2 0.012 0.32 0.0036 0.052 0.022 -0.033
mainroad 0.28 0.25 -0.013 0.0088 0.13 1 0.081 0.048 0.032 0.11 0.17 0.17 0.058 -0.11
guestroom 0.25 0.25 0.029 0.17 -0.019 0.081 1 0.35 0.024 0.11 0.025 0.22 0.043 -0.12
basement 0.2 0.11 0.1 0.12 -0.2 0.048 0.35 1 0.0034 0.07 0.09 0.27 0.035 -0.12
hotwaterheating 0.059 -0.025 -0.0066 0.046 0.012 0.032 0.024 0.0034 1 -0.1 0.063 -0.067 0.096 -0.071
airconditioning 0.47 0.26 0.15 0.2 0.32 0.11 0.11 0.07 -0.1 1 0.12 0.11 -0.01 -0.12
parking 0.32 0.33 0.13 0.15 0.0036 0.17 0.025 0.09 0.063 0.12 1 0.038 0.065 -0.13
prefarea 0.33 0.21 0.087 0.029 0.052 0.17 0.22 0.27 -0.067 0.11 0.038 1 -0.037 -0.087
semi-furnished 0.13 0.053 0.098 0.039 0.022 0.058 0.043 0.035 0.096 -0.01 0.065 -0.037 1 -0.63
unfurnished -0.28 -0.12 -0.14 -0.12 -0.033 -0.11 -0.12 -0.12 -0.071 -0.12 -0.13 -0.087 -0.63 1
price
area
stories
parking
prefarea

FNNs

Advantages:

- Simple to understand and implement.
- Works well for structured data (e.g., tabular data).

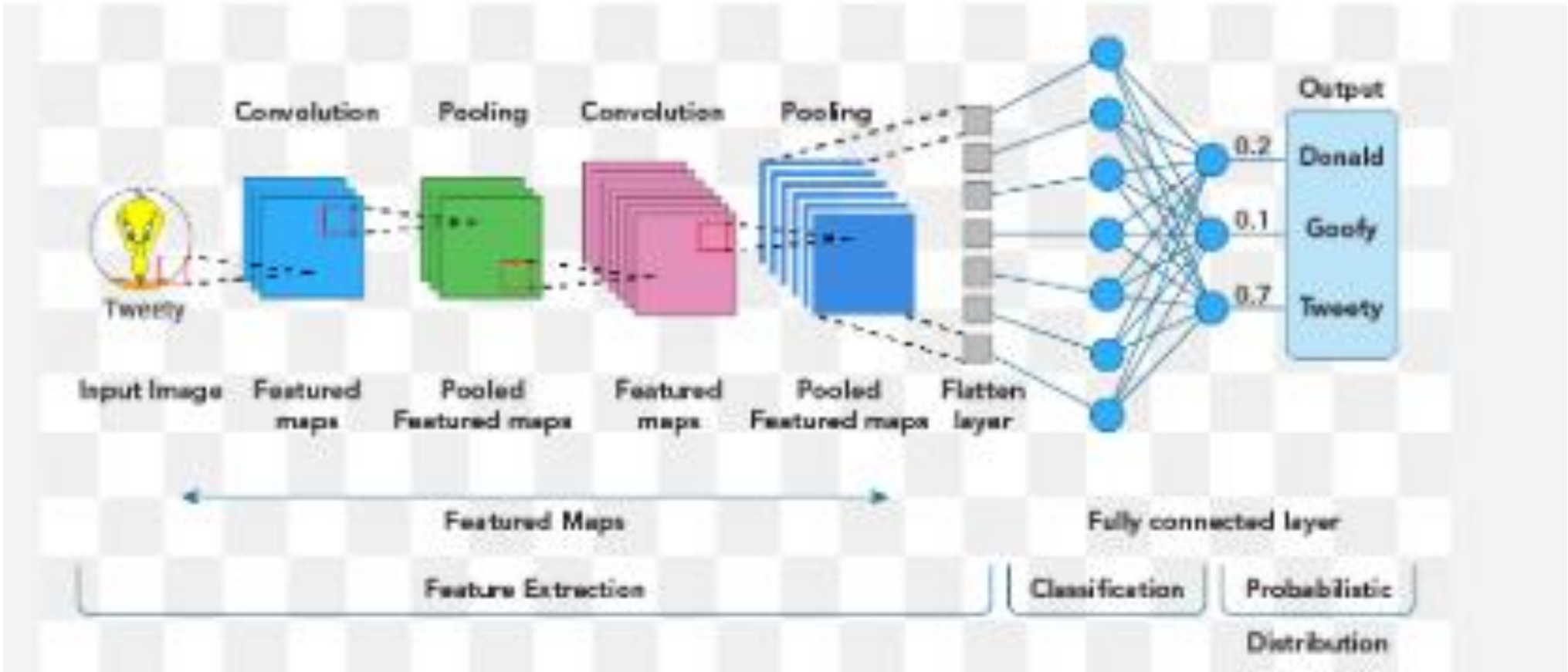
Limitations:

- Cannot capture temporal dependencies or patterns in sequential data (like text or time series).

Tabular Data

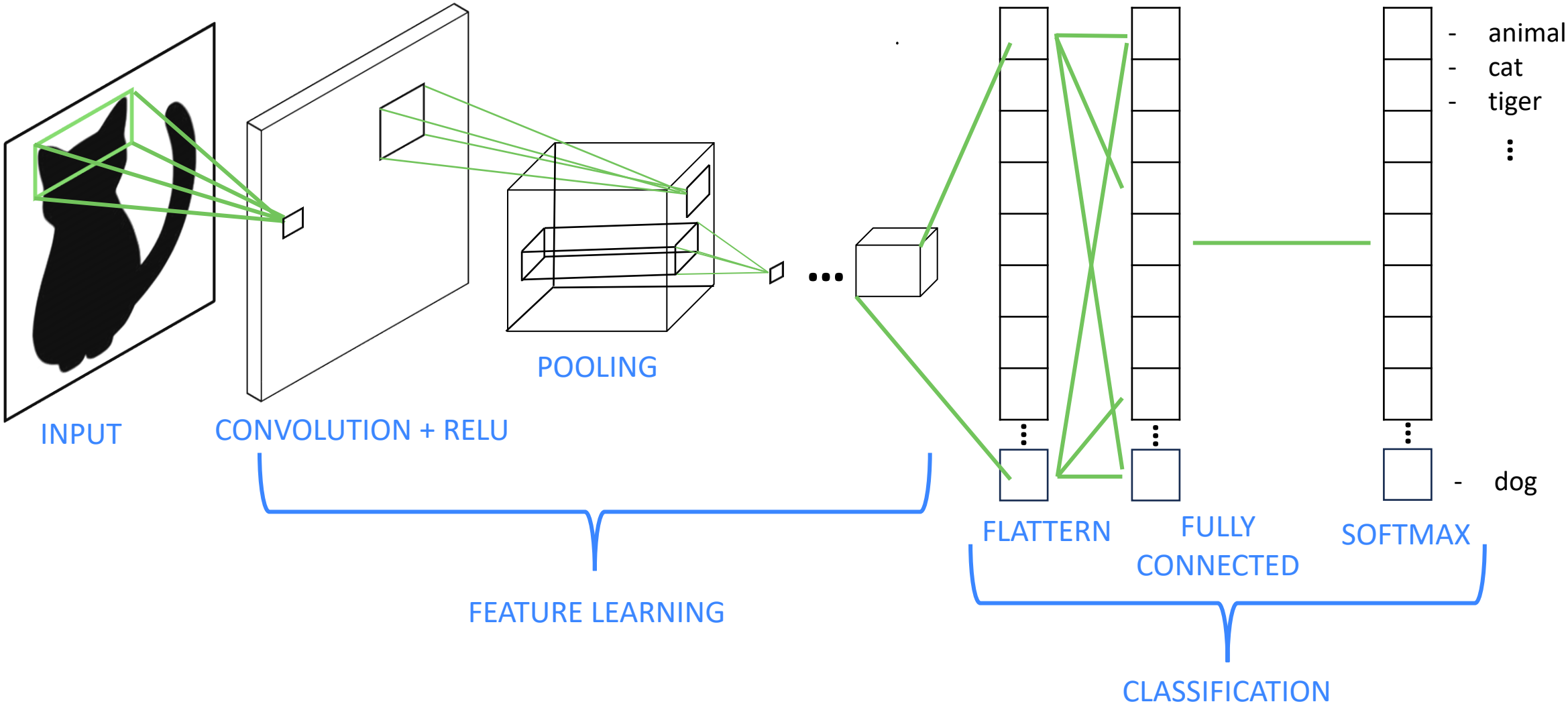
Patients		
Name	Gender	Age
Bill	Male	21
Miko	Female	15
Juan	Other	45

Convolutional Neural Networks (CNNs)


Convolution
Pooling
Convolution
Pooling
Output
0.2
Donald
0.1
Goofy
0.7
Tweety
Tweety
Input Image
Featured maps
Pooled Featured maps
Featured maps
Pooled Featured maps
Flatten layer
Featured Maps
Fully connected layer
Feature Extraction
Classification
Probabilistic Distribution

Convolutional Neural Networks (CNNs)

Definition: A specialized type of neural network designed for **image and spatial data** processing.



CNNs

Characteristics

- Uses **convolutional layers** that apply filters to detect patterns such as edges, textures, or objects.
- Consists of additional layers like **Pooling Layers** (to reduce dimensionality) and **Fully Connected Layers** (for classification).

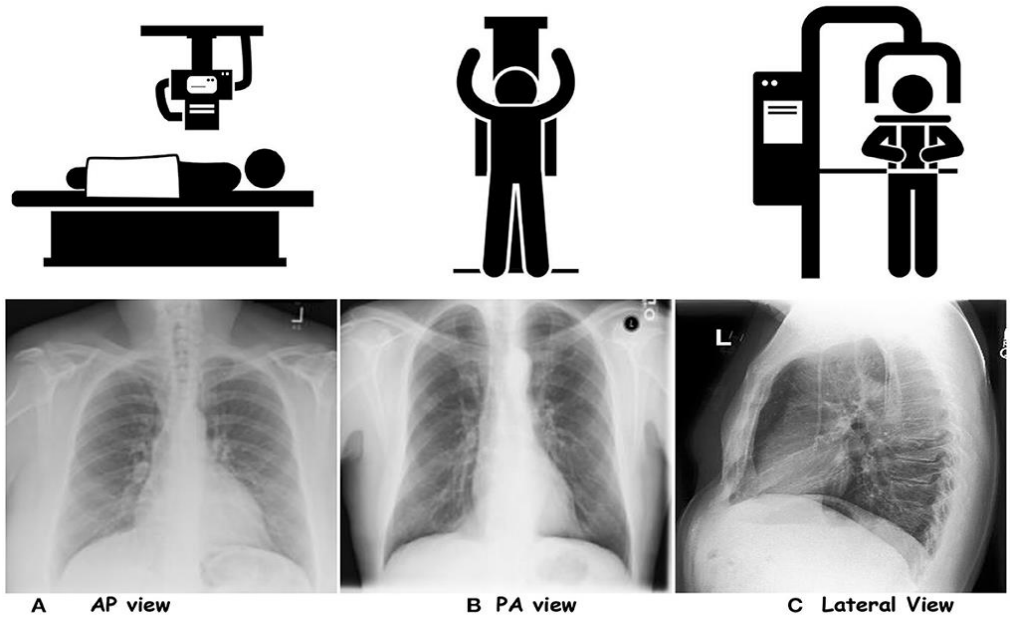
Key Components

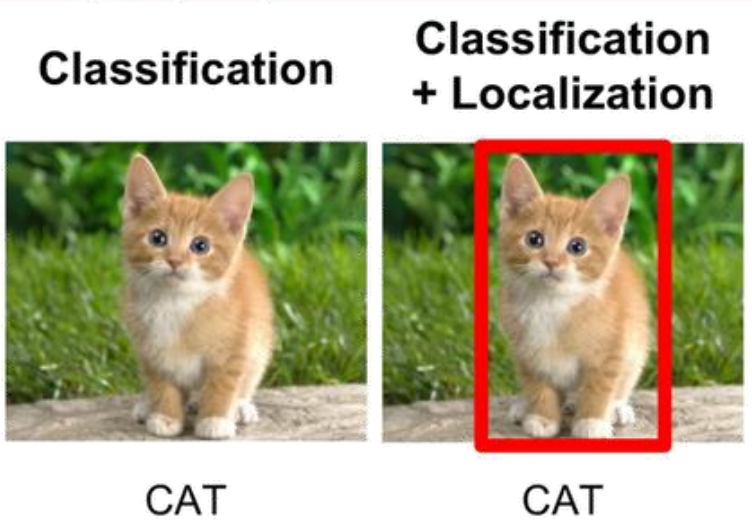
- **Convolutional Layer**: Applies a set of filters to the input image to extract features.
- **Pooling Layer**: Reduces the size of the feature maps, preserving important features while reducing computation.
- **Fully Connected Layer**: Connects neurons from the previous layers to produce the final output.

CNNs

Applications

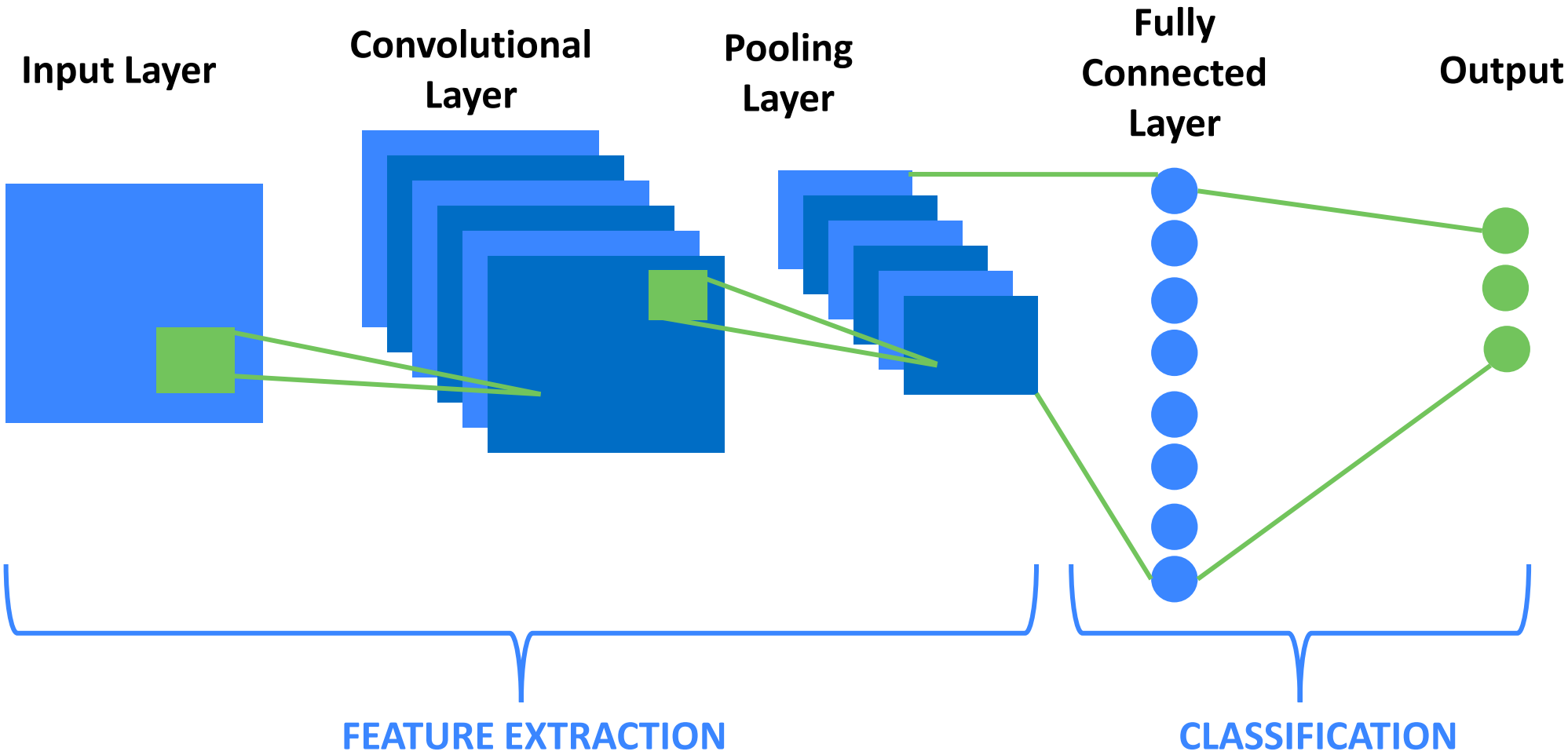
- Image classification (e.g., identifying objects in photos).
- Facial recognition.
- Medical imaging (e.g., detecting tumors in X-rays).





CNNs

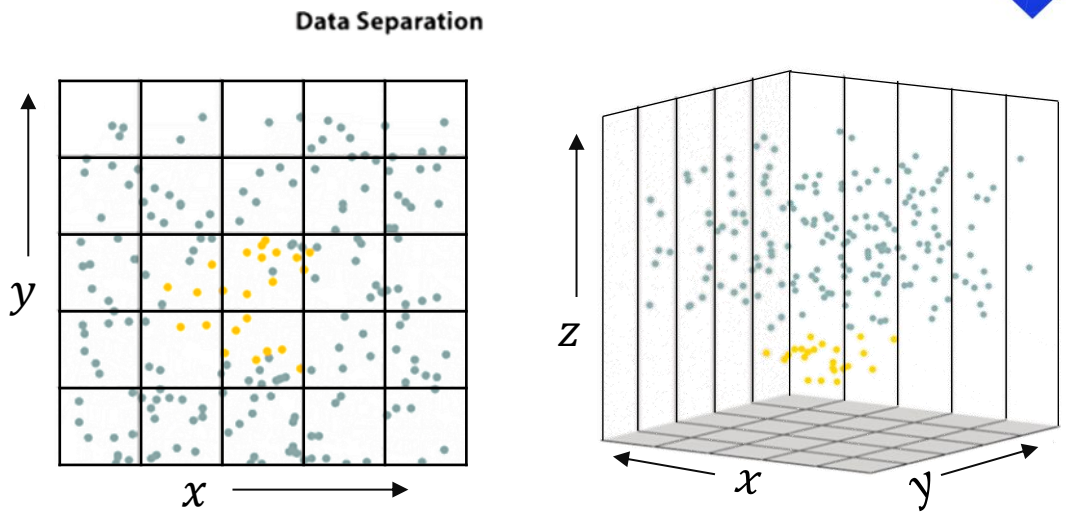
- Input Image → Convolutional Layer → Pooling Layer → Fully Connected Layer → Output Layer



CNNs

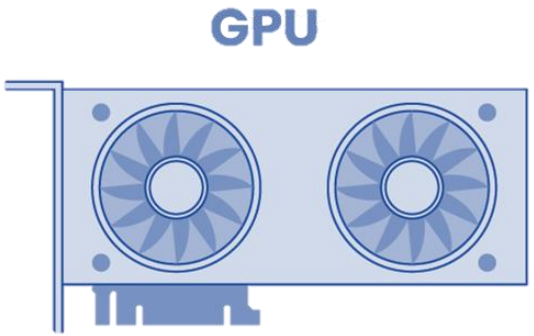
Advantages:

- Excellent at detecting spatial hierarchies in images.
- Efficient in processing high-dimensional data like images.



Limitations:

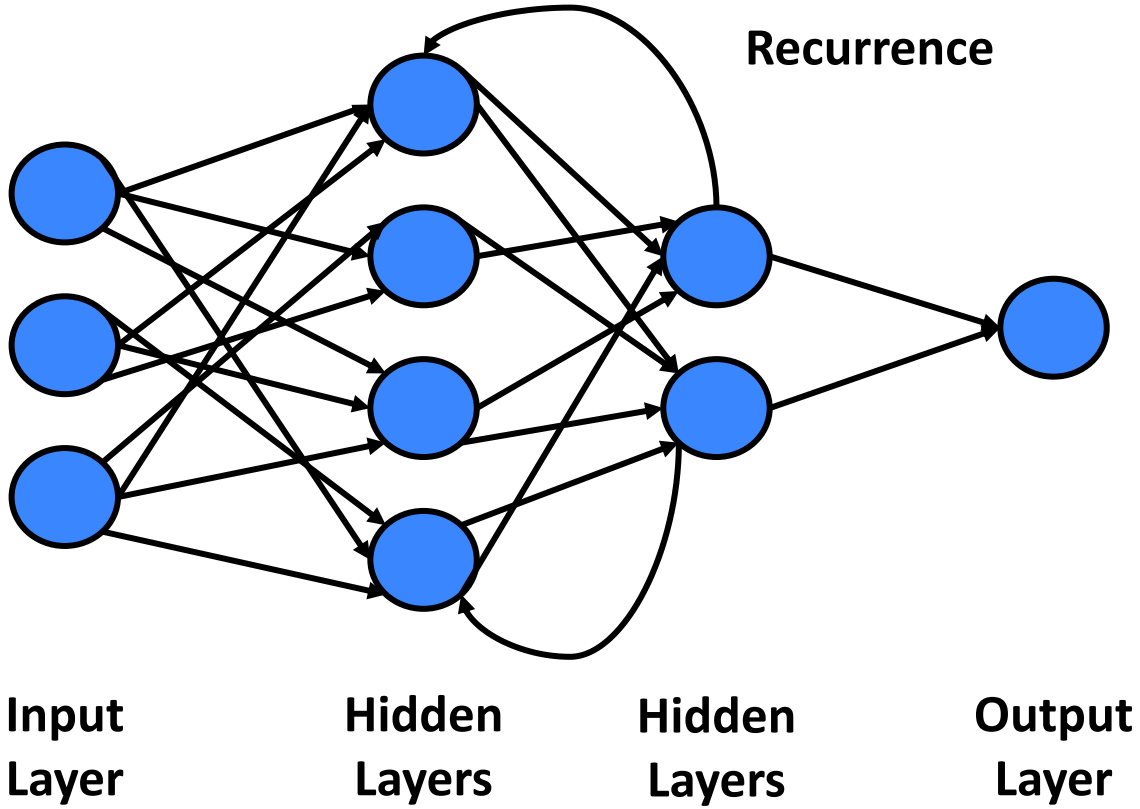
- Requires a lot of data for training.
- Computationally intensive and may require specialized hardware (e.g., GPUs).



Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs)

Definition: A type of neural network designed to process **sequential data** by retaining information from previous inputs.



RNNs

Characteristics

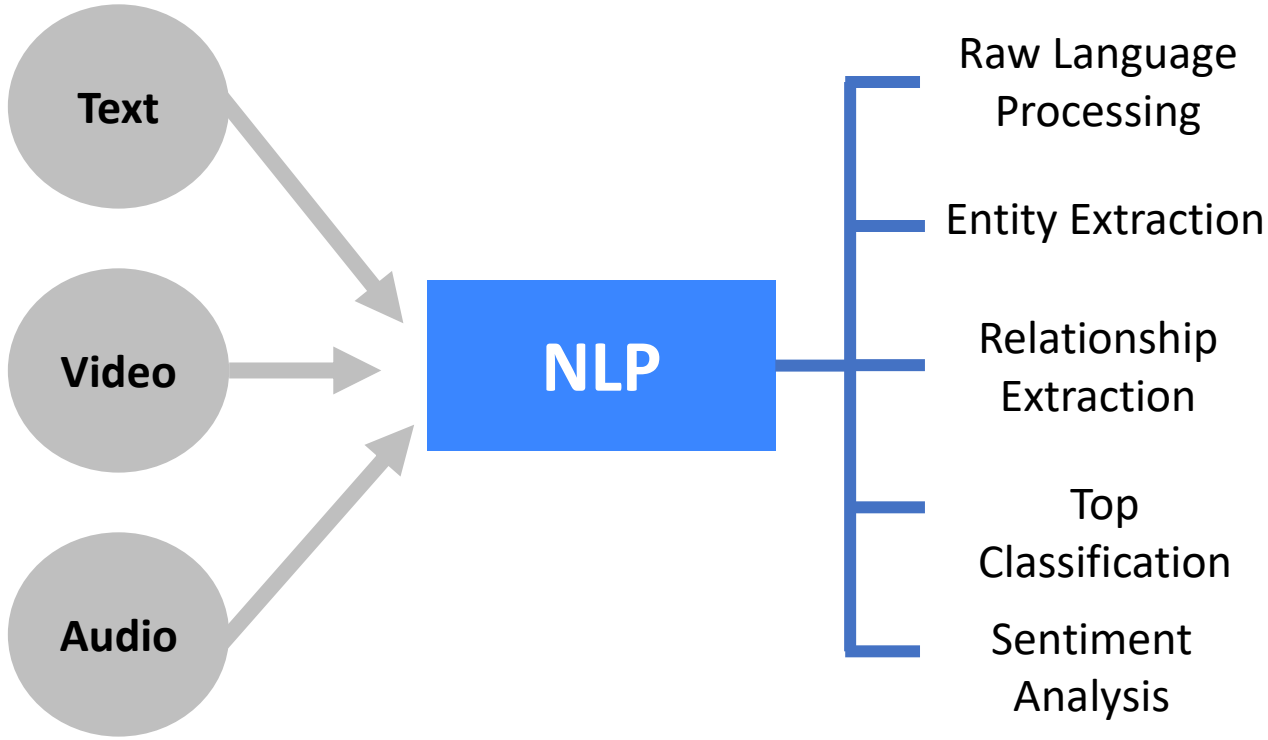
- Unlike feedforward networks, RNNs have connections that form **loops**, allowing them to maintain a **memory** of previous inputs.
- Uses **hidden states** to carry information across time steps.

Key Components

- **Hidden State**: Maintains information from previous time steps to influence future outputs.
- **Recurrent Layers**: Share parameters across time steps, enabling the model to process sequences of arbitrary length.

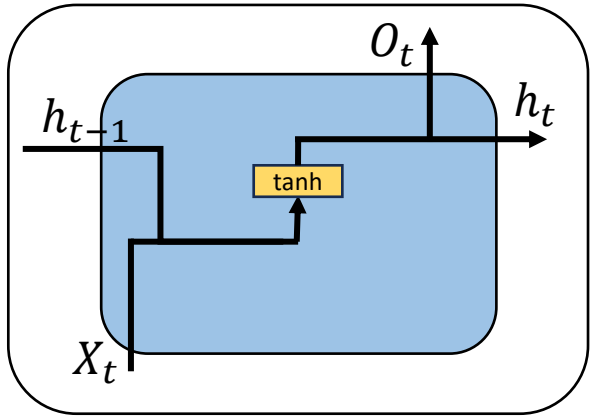
Applications

- Natural Language Processing (NLP): Text generation, language translation, sentiment analysis.
- Time Series Analysis: Stock price prediction, weather forecasting.
- Speech Recognition: Converting spoken words into text.

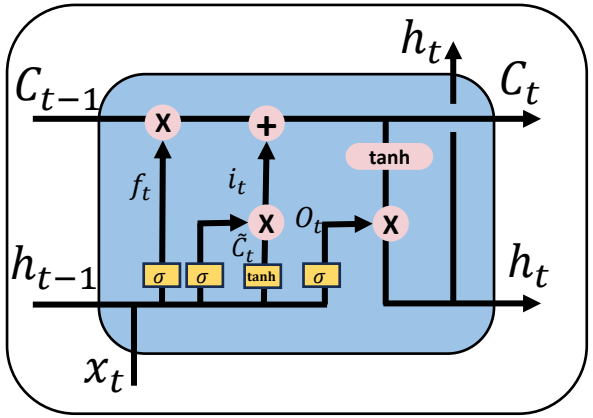


Example Architecture

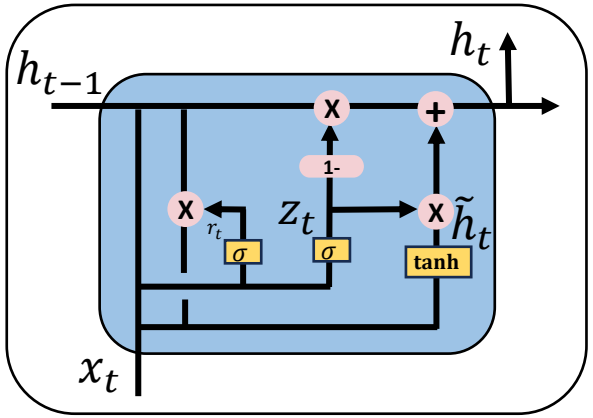
- Sequence Input → Recurrent Layer (e.g., LSTM or GRU) → Output Layer



RNN



LSTM



GRU

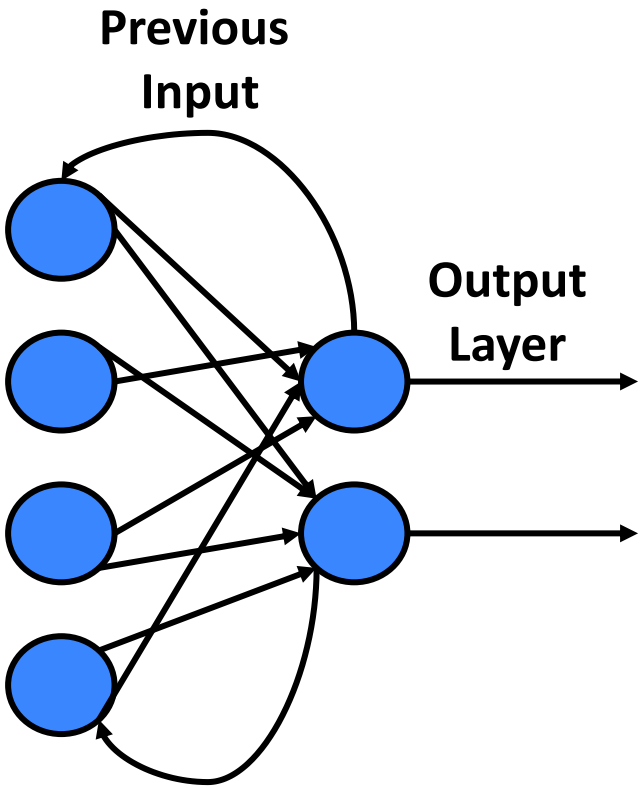
RNNs

Advantages:

- Great at handling temporal dependencies in sequences.
- Useful for applications where context from previous inputs is crucial.

Limitations:

- Difficult to train due to issues like vanishing and exploding gradients.
- Slower to train compared to feedforward networks due to sequential data processing.



Summary

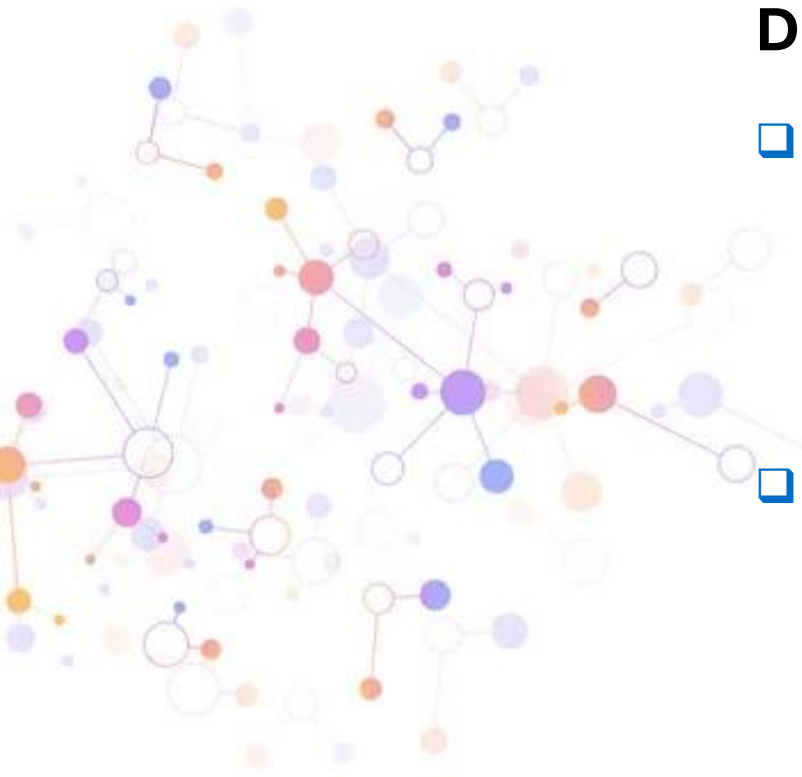
Type	Suitable For	Example Applications
Feedforward Neural Networks	Structured data, basic classification	Spam detection, regression
Convolutional Neural Networks	Image and spatial data	Object detection, medical imaging
Recurrent Neural Networks	Sequential data	Language translation, stock prediction

- Neural networks consist of an **input layer**, **hidden layers**, and an **output layer**.
- **Feedforward Neural Networks** are the simplest form and are ideal for structured data.
- **Convolutional Neural Networks** are specialized for processing images and spatial data.
- **Recurrent Neural Networks** excel at handling sequential data, such as text and time series.

Neural Network Architecture

This section focuses on the architecture of neural networks, including the key parameters that define it, and how the network is trained through forward propagation, backpropagation, and gradient descent. Understanding these concepts is essential for designing and optimizing neural networks.

What is an Architecture?



Definition:

- **Neural Network Architecture** refers to the structure or design of a neural network, which includes the arrangement of its **layers**, the number of **neurons** in each layer, the **connections** between neurons, and the **activation functions** used.
- Just like the blueprint of a building, the architecture of a neural network determines how it processes inputs, transforms data, and produces outputs.

Neural Network Architecture

Key Points

- A well-designed architecture can efficiently solve complex problems, while a poorly designed one may lead to suboptimal performance.
- The architecture is tailored to the **type of problem** you are trying to solve (e.g., image classification, text generation, time series forecasting).

Analogy

- Think of the neural network architecture as a recipe. The **layers, neurons, and activation functions** are the ingredients, while the **training process** (forward and backward propagation) is like following the cooking instructions to create the final dish.

Neural Network Architecture

Creating the Neural Network in Excel

Using Excel to simulate the basic workings of a neural network, including inputs, weights, summation, activation functions, and outputs. The neural network is designed to predict an outcome based on three input features from your training dataset, as illustrated in the table below.

Input 1	Input 2	Input 3	Output
1	0	1	1
1	1	0	1
0	0	0	0
1	1	1	1
1	0	0	1
0	1	0	0
1	0	1	1
0	1	1	0

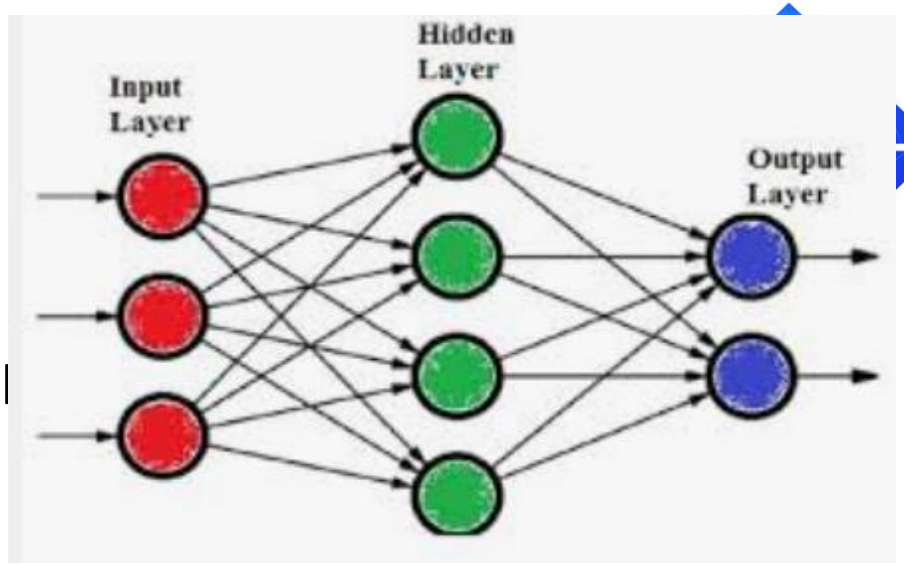
Neural Network Architecture

Step 1 – Input Data

Begin by opening a new Excel spreadsheet and inputting your data, which will serve as the training dataset for your neural network. This dataset will be used to teach the model how to make predictions based on the relationships within the data.

	A	B	C	D
1	Training Data			
2	Input 1	Input 2	Input 3	Output
3	1	0	1	1
4	1	1	0	1
5	0	0	0	0
6	1	1	1	1
7	1	0	0	1
8	0	1	0	0
9	1	0	1	1
10	0	1	1	0

Neural Network Architecture



Step 2 – Set Up Weights and Biases

Weights and biases are critical in neural networks to adjust t
different inputs (features) and improve predictions.

- **Weights:** Values that the model adjusts during training to give different levels of importance to different features.
- **Biases:** A baseline or extra bit of information added to the prediction.

Together, weights and biases enable the neural network to adjust its predictions by emphasizing the significance of different factors and identifying patterns within the data. Think of them as **the adjustable parameters** that the model fine-tunes during the training process to enhance its accuracy.

Neural Network Architecture

- Starting in the next empty column, create weight columns for each of the three inputs: “Weight 1,” “Weight 2,” and “Weight 3.”
- Create a “Bias” column.
- Calculate the weights and bias for the first record in your spreadsheet. To get started with this example, you can input random values for your weights: 72 percent, 77 percent, and 85 percent.

	A	B	C	D	E	F	G	H
1	Training Data				Weights			
2	Input 1	Input 2	Input 3	Output	Weight 1	Weight 2	Weight 3	Bias
3	1	0	1	1	72%	77%	85%	
4	1	1	0	1				
5	0	0	0	0				
6	1	1	1	1				
7	1	0	0	1				
8	0	1	0	0				
9	1	0	1	1				
10	0	1	1	0				

Neural Network Architecture

- To determine the **Bias** column values, calculate the weighted sum of all your inputs by multiplying each input by its corresponding weight and adding up the results. You can use the following Excel formula to do this:

  (E3*A3) + (F3*B3) + (G3*C3)

- Insert the formula into the first row's Bias field. Your spreadsheet should look like this Figure.

	A	B	C	D	E	F	G	H
1	Training Data				Weights			
2	Input 1	Input 2	Input 3	Output	Weight 1	Weight 2	Weight 3	Bias
3	1	0	1	1	72%	77%	85%	1.57
4	1	1	0	1				
5	0	0	0	0				
6	1	1	1	1				
7	1	0	0	1				
8	0	1	0	0				
9	1	0	1	1				
10	0	1	1	0				

Neural Network Architecture

Step 3 – Calculate Your Output

Next, calculate the **output** value using the Sigmoid Function to normalize the bias. This function helps the network make more balanced predictions by capturing complex data relationships non-linearly. The Sigmoid is represented by the equation:

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

Neural Network Architecture

- To the right of the Bias column, create a column to hold your predictions and label it "**Output.**"
- Use the following Excel formula to calculate your output: =1/(1+(EXP(-H3)))
- Insert this formula into your first record's Output field. Your spreadsheet should resemble like shown below.

	A	B	C	D	E	F	G	H	I
1		Training Data				Weights			
2	Input 1	Input 2	Input 3	Output	Weight 1	Weight 2	Weight 3	Bias	Output
3	1	0	1	1	72%	77%	85%	1.57	0.827784
4	1	1	0	1					
5	0	0	0	0					
6	1	1	1	1					
7	1	0	0	1					
8	0	1	0	0					
9	1	0	1	1					
10	0	1	1	0					

- The first row's output (0.827783608) doesn't match the actual value (1) in your training data, indicating the neural network needs fine-tuning for better accuracy.

Neural Network Architecture

Step 4 – Fine Tune Your Weights

In this step, you'll adjust the **weights** to improve accuracy—think of weights as dials that you can fine-tune to enhance the model's predictions.

- Add three new columns labeled “Weight 1 Δ,” Weight 2 Δ,” and “Weight 3 Δ.” These will hold the change values for applying to your initial weights.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Training Data				Weights					Weight Tuning		
2	Input 1	Input 2	Input 3	Output	Weight 1	Weight 2	Weight 3	Bias	Output	Weight 1 Δ	Weight 2 Δ	Weight 3 Δ
3	1	0	1	1	72%	77%	85%	1.57	0.827784			
4	1	1	0	1								
5	0	0	0	0								
6	1	1	1	1								
7	1	0	0	1								
8	0	1	0	0								
9	1	0	1	1								
10	0	1	1	0								

Neural Network Architecture

To calculate the weight adjustments, you'll use the error-weighted derivative formula through backpropagation, where weights are updated from right to left. For 'Weight 1 Δ,' the formula in Excel is:

=($D3-$I3)*A3*$I3*(1-$I3)

This

netv

	A	B	C	D	E	F	G	H	I	J	K	L
1	Training Data				Weights					Weight Tuning		
2	Input 1	Input 2	Input 3	Output	Weight 1	Weight 2	Weight 3	Bias	Output	Weight 1 Δ	Weight 2 Δ	Weight 3 Δ
3	1	0	1	1	72%	77%	85%	1.57	0.827784			
4	1	1	0	1								
5	0	0	0	0								
6	1	1	1	1								
7	1	0	0	1								
8	0	1	0	0								
9	1	0	1	1								
10	0	1	1	0								

Neural Network Architecture

- Paste the formula in the “Weight 1 Δ” cell.
- After calculating the value for “Weight 1 Δ,” drag the formula across to “Weight 2 Δ” and “Weight 3 Δ” to calculate their values as well. Your spreadsheet should now look like an image shown below.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Training Data				Weights					Weight Tuning		
2	Input 1	Input 2	Input 3	Output	Weight 1	Weight 2	Weight 3	Bias	Output	Weight 1 Δ	Weight 2 Δ	Weight 3 Δ
3	1	0	1	1	72%	77%	85%	1.57	0.827784	0.02455081	0	0.02455081
4	1	1	0	1								
5	0	0	0	0								
6	1	1	1	1								
7	1	0	0	1								
8	0	1	0	0								
9	1	0	1	1								
10	0	1	1	0								

Neural Network Architecture

Next, you'll apply the new weight tuning values to the second row of your training data.

Insert the following Excel formula into the second row's "Weight 1" field: = J3+E3

Drag the formula across to the second row's "Weight 2" and "Weight 3" fields, as well. Your spreadsheet should look like Figure below.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Training Data				Weights					Weight Tuning		
2	Input 1	Input 2	Input 3	Output	Weight 1	Weight 2	Weight 3	Bias	Output	Weight 1 Δ	Weight 2 Δ	Weight 3 Δ
3	1	0	1	1	72%	77%	85%	1.57	0.827784	0.02455081	0	0.02455081
4	1	1	0	1	74%	77%	87%					
5	0	0	0	0								
6	1	1	1	1								
7	1	0	0	1								
8	0	1	0	0								
9	1	0	1	1								
10	0	1	1	0								

Neural Network Architecture

Step 5 – Calculate Remaining Values

Next, calculate the bias, output, and weight adjustment values for the other rows.

- Drag the Bias (cell H3), Output (cell I3), and Weight Tuning (cells J3, K3, L3) calculations down to the last row so your spreadsheet looks like Figure shown below.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Training Data				Weights					Weight Tuning		
2	Input 1	Input 2	Input 3	Output	Weight 1	Weight 2	Weight 3	Bias	Output	Weight 1 Δ	Weight 2 Δ	Weight 3 Δ
3	1	0	1	1	72%	77%	85%	1.57	0.827784	0.02455081	0	0.02455081
4	1	1	0	1	74%	77%	87%	1.514551	0.819735	0.02663776	0.02663776	0
5	0	0	0	0				0	0.5	0	0	0
6	1	1	1	1				0	0.5	0.125	0.125	0.125
7	1	0	0	1				0	0.5	0.125	0	0
8	0	1	0	0				0	0.5	0	-0.125	0
9	1	0	1	1				0	0.5	0.125	0	0.125
10	0	1	1	0				0	0.5	0	-0.125	-0.125

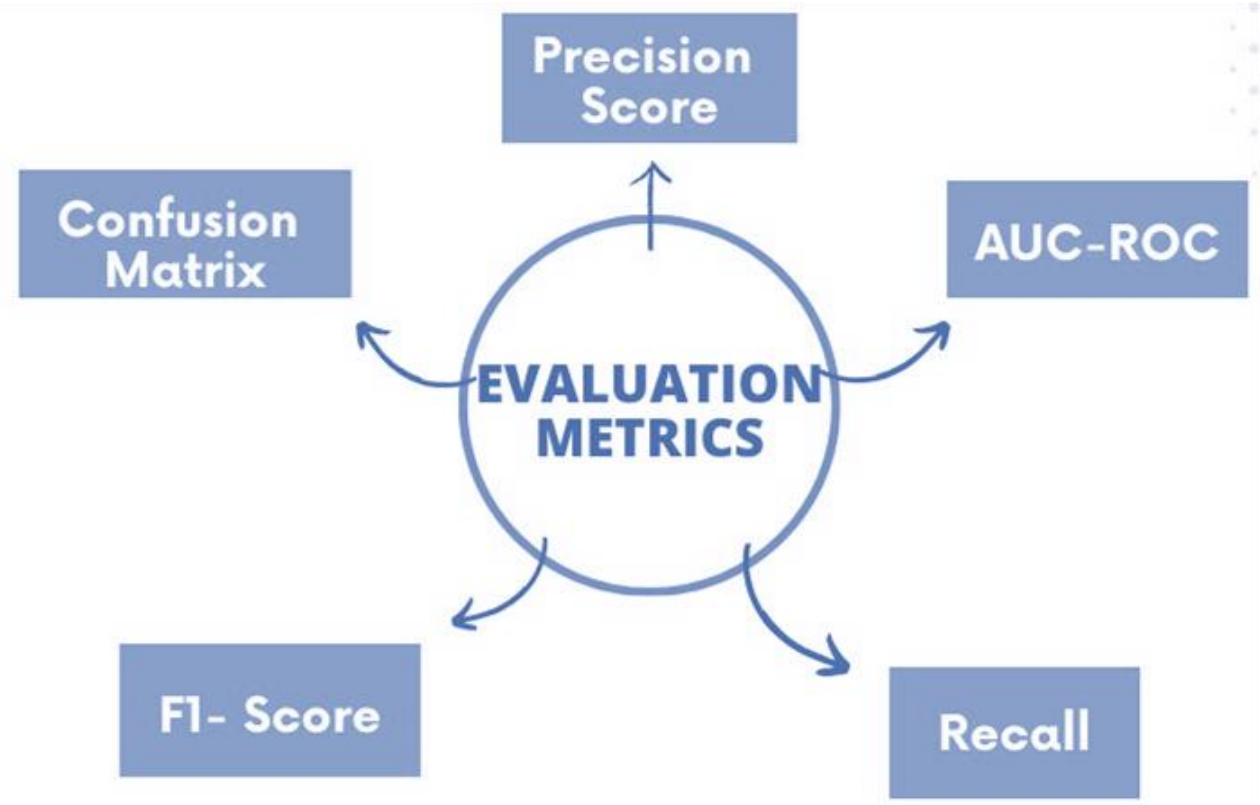
Neural Network Architecture

- Now that you have all your weight tuning calculations in place, calculate the rest of your weight values (e.g., Weight 1, Weight 2, Weight 3) by dragging them down to the last row.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Training Data				Weights					Weight Tuning		
2	Input 1	Input 2	Input 3	Output	Weight 1	Weight 2	Weight 3	Bias	Output	Weight 1 Δ	Weight 2 Δ	Weight 3 Δ
3	1	0	1	1	72%	77%	85%	1.57	0.827784	0.02455081	0	0.02455081
4	1	1	0	1	74%	77%	87%	1.514551	0.819735	0.02663776	0.02663776	0
5	0	0	0	0	77%	80%	87%	0	0.5	0	0	0
6	1	1	1	1	77%	80%	87%	2.442377	0.920002	0.00588769	0.00588769	0.00588769
7	1	0	0	1	78%	80%	88%	0.777076	0.68505	0.06795263	0	0
8	0	1	0	0	85%	80%	88%	0.802525	0.690514	0	-0.1475659	0
9	1	0	1	1	85%	65%	88%	1.725467	0.848832	0.01939737	0	0.01939737
10	0	1	1	0	86%	65%	90%	1.554795	0.825605	0	-0.1188716	-0.1188716

Evaluating Model Performance

Different metrics are used to assess the quality of a neural network model, depending on the problem type (classification, regression, etc.). Here, we'll focus on the most common metrics used for classification tasks.



Model Evaluation : Confusion Metrics

- In classification problems, the primary source for accuracy estimation is the confusion matrix:

		Actual Result	
		Positive	**Negative**
Predictive/ Classification Result	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

TP = number of samples predicted positive that are actually positive
FP = number of samples predicted positive that are actually negative
TN = number of samples predicted negative that are actually negative
FN = number of samples predicted negative that are actually positive

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$True\ Positive\ Rate = \frac{TP}{TP + FN}$$

$$True\ Negative\ Rate = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP}$$

Evaluating Model Performance

a) Accuracy

- **Definition**: The proportion of correctly predicted labels to the total number of predictions.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

- **Use Case**: Best used when the dataset is balanced (i.e., the classes are evenly distributed).

Evaluating Model Performance

b) Loss

- **Definition**: A measure of how well the model's predictions match the true labels.

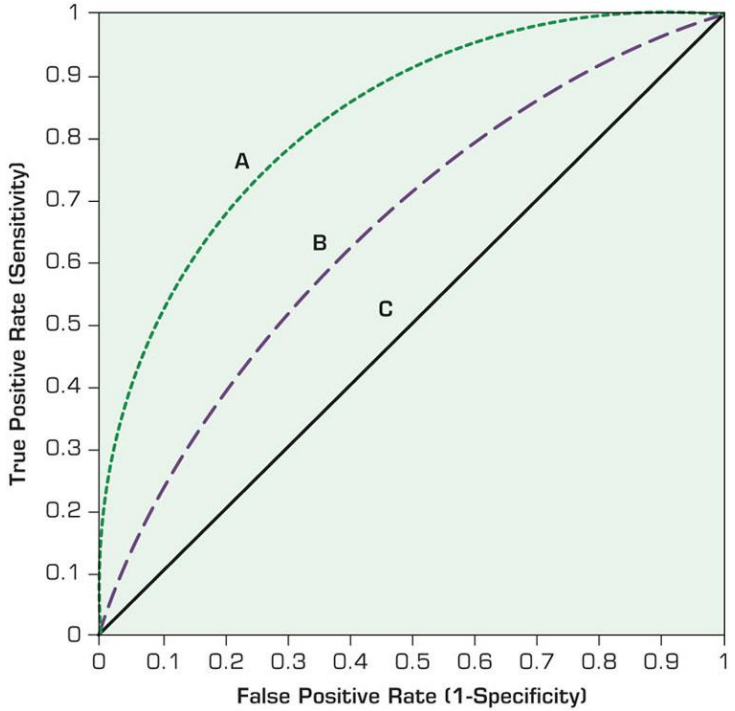
  The loss function is minimized during training.

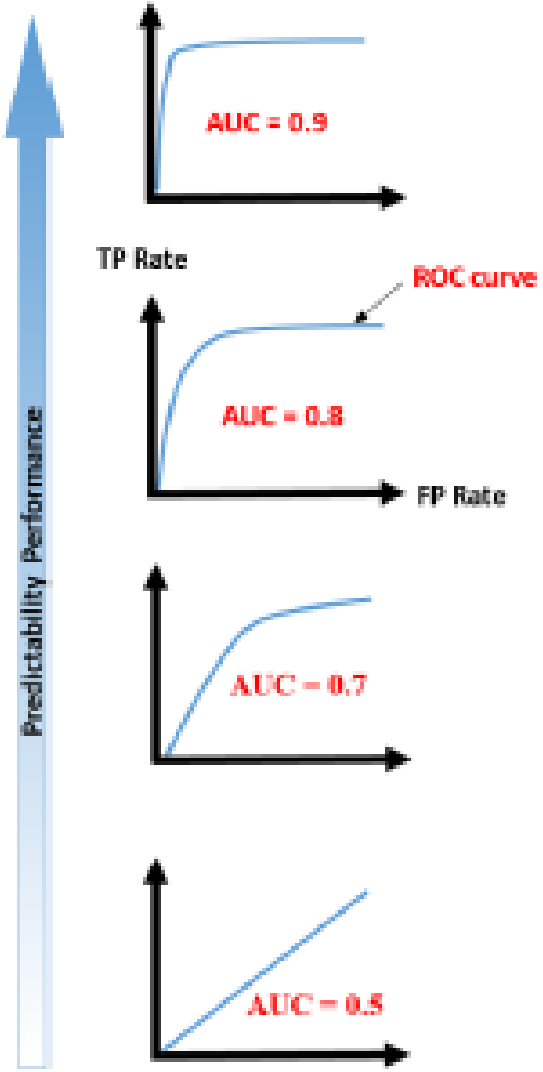
  - Common loss functions:
    - **Cross-Entropy Loss**: Used for classification problems.
    - **Mean Squared Error (MSE)**: Used for regression problems.

- **Use Case:** Lower loss indicates better model performance, but it should be complemented by other metrics.

Model Evaluation Metrics : AUC-ROC

- Area Under the ROC Curve (AUC) is a single number that captures the overall quality of a classifier. It should be between 0.5 (random classifier) and 1.0 (perfect).





An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds

Evaluating Model Performance

c) Precision

- **Definition:** The proportion of true positive predictions to the total number of positive predictions made by the model.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

- **Use Case:** Useful when **false positives** are costly (e.g., email spam filtering).

Evaluating Model Performance

d) Recall (Sensitivity or True Positive Rate)

- **Definition:** The proportion of true positive predictions to the total actual positives in the dataset.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

- **Use Case:** Important when **false negatives** are costly (e.g., cancer detection).

Evaluating Model Performance

e) F1 Score

- **Definition:** The harmonic mean of precision and recall, providing a balance between the two.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- **Use Case:** Useful when you need a balance between precision and recall, especially in **imbalanced datasets**.

Evaluating Model Performance

Example Summary of Metrics:

Metric	*Best for*	*When to Use*
Accuracy	Balanced datasets	General performance measurement
Precision	False positives are costly	Spam detection, fraud detection
Recall	False negatives are costly	Medical diagnosis, safety-critical systems
F1 Score	Imbalanced datasets	When both precision and recall are important

Realistic Infotech Group
IT Training & Services
No.79/A, First Floor
Corner of Insein Road and
Damaryon Street
Quarter (9), Hlaing Township
Near Thukha Bus Station
09256675642, 09953933826
http://www.rig-info.com