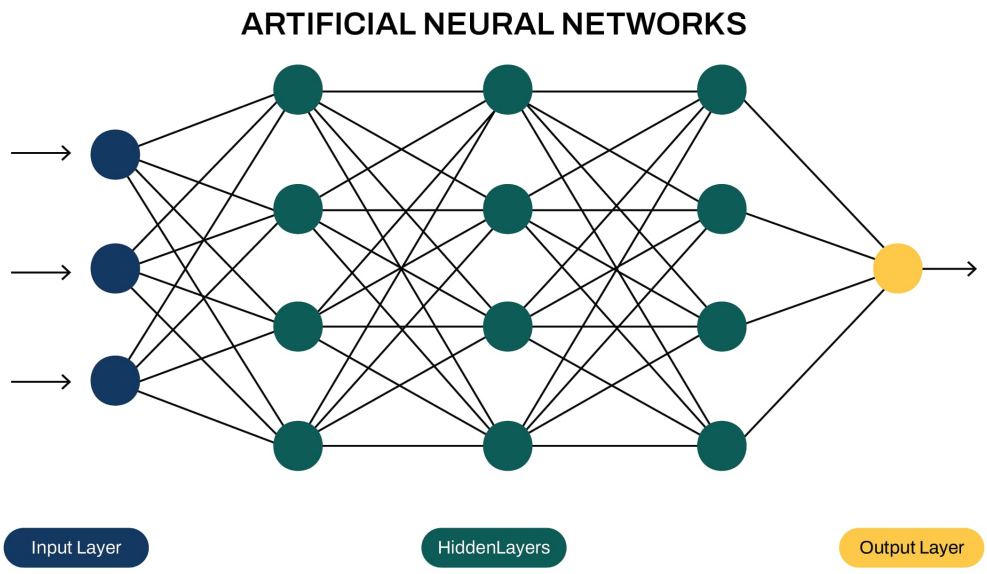
Neural Network Architecture:




1. **Input Layer:** The input layer is responsible for receiving the initial data, which could be numeric, textual, or image-based. The number of neurons in the input layer corresponds to the dimensions of the input data.
2. **Hidden Layers:** Hidden layers are intermediate layers between the input and output layers. They enable the neural network to learn complex patterns and representations. The number of hidden layers and the number of neurons in each layer depend on the complexity of the problem and the available data.
3. **Output Layer:** The output layer produces the final result or prediction based on the learned patterns. The number of neurons in the output layer depends on the problem type, such as binary classification, multi-class classification, or regression.

To build a neural network in Python, we can follow these key steps:

1. **Data Preparation:** Gather and preprocess the data for training and evaluation, split the data into training and testing sets, and ensuring an adequate representation of the problem domain.
2. **Model Architecture:** Determine the number of input features and the desired output shape, choose the number of hidden layers and the number of neurons in each layer and select an appropriate activation function for each layer. Activation functions introduce non-linearity into the neural network, enabling it to learn complex relationships and make predictions. Some commonly used activation functions include:
   **a. Sigmoid:** The sigmoid function maps the input to a value between 0 and 1, making it suitable for binary classification problems.
   **b. ReLU (Rectified Linear Unit):** The ReLU function sets all negative inputs to zero and keeps positive inputs unchanged. It is widely used in hidden layers to introduce non-linearity and address the vanishing gradient problem.
   **c. Softmax:** The softmax function is commonly used in the output layer for multi-class classification tasks. It converts the outputs into probabilities, ensuring they sum up to 1.

3. **Model Compilation:** Specify the loss function, which measures the model's performance, choose an optimization algorithm, such as stochastic gradient descent (SGD), to update the model's weights during training, and Define evaluation metrics to measure the model's accuracy or performance.
4. **Model Training:** Feed the training data into the neural network and iterate through multiple epochs, adjust the model's weights using backpropagation, which calculates the gradients of the loss function with respect to the weights and optimize the model by minimizing the loss function and improving its performance.
5. **Model Evaluation:** Assess the model's performance using the testing data, calculate evaluation metrics, such as accuracy, precision, recall, or F1 score, to measure its effectiveness and iterate and fine-tune the model based on the evaluation results.