Problem Statement

You want to build an ANN that can classify emails into two categories: "Spam" and "Not Spam." You have a dataset of emails where each email is labeled as either spam or not spam.

Input Features

Let's consider a simplified scenario with only three features for each email:

1. **Feature 1 (X1):** Number of words related to advertisements (e.g., "buy," "free," "offer").
2. **Feature 2 (X2):** Number of exclamation marks.
3. **Feature 3 (X3):** Number of suspicious links.

These features will be the inputs to our neural network.

ANN Structure

We'll use a simple feedforward neural network with the following structure:

- **Input Layer:** 3 neurons (one for each feature)
- **Hidden Layer:** 4 neurons (This is arbitrary, but it's a common practice to start with a small hidden layer for a simple problem)
- **Output Layer:** 1 neuron (for binary classification, with an activation function like sigmoid to produce a probability)

Step 1: Initialization

1. **Weights:** Initialize small random weights for each connection between layers.
2. **Biases:** Initialize small random biases for the neurons in the hidden and output layers.

Step 2: Forward Propagation

1. **Input Layer:** Each feature (X1, X2, X3) is fed into the respective input neurons.
2. **Hidden Layer:**
   - Each neuron in the hidden layer calculates a weighted sum of the inputs:
     $$Z_i^{(1)} = W_{i1}^{(1)} \cdot X_1 + W_{i2}^{(1)} \cdot X_2 + W_{i3}^{(1)} \cdot X_3 + b_i^{(1)}$$
     where $Z_i^{(1)}$ is the weighted sum for the i^{th} neuron in the hidden layer, $W_{ij}^{(1)}$ are the weights, and $b_i^{(1)}$ is the bias for the i^{th} hidden neuron.
   - The weighted sum is passed through an activation function, such as ReLU (Rectified Linear Unit), to introduce non-linearity:
     $$A_i^{(1)} = \text{ReLU}(Z_i^{(1)})$$
3. **Output Layer:**
   - The output neuron calculates a weighted sum of the hidden layer's activations:
     $$Z^{(2)} = W_1^{(2)} \cdot A_1^{(1)} + W_2^{(2)} \cdot A_2^{(1)} + W_3^{(2)} \cdot A_3^{(1)} + W_4^{(2)} \cdot A_4^{(1)} + b^{(2)}$$
   - This sum is passed through a sigmoid activation function to produce the final output (a probability):

$$\hat{y} = \sigma(Z^{(2)}) = \frac{1}{1 + e^{-Z^{(2)}}}$$

- If $\hat{y} \geq 0.5$, classify the email as "Spam." Otherwise, classify it as "Not Spam."

Step 3: Loss Calculation

- **Loss Function:** Use a binary cross-entropy loss function to compare the predicted output ($\hat{y}$) with the true label (y):

$$\text{Loss} = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})]$$

Step 4: Backpropagation

- **Gradient Calculation:** Compute the gradient of the loss function with respect to each weight and bias in the network.
- **Weight Update:** Adjust the weights and biases using gradient descent or another optimization algorithm:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \cdot \frac{\partial \text{Loss}}{\partial W_{ij}^{(l)}}$$

  where α is the learning rate.

Step 5: Iteration and Training

- **Iterate:** Repeat the forward and backward propagation for many epochs (iterations) over the entire dataset.
- **Training:** As the network trains, it learns the optimal weights and biases to minimize the loss and improve accuracy.

Step 6: Making Predictions

- After training, the ANN can predict whether a new email is "Spam" or "Not Spam" by feeding it through the network and observing the output.

Visualization

Imagine this process as follows:

- **Input Layer:** Three nodes representing X1, X2, and X3.
- **Hidden Layer:** Four nodes connected to all input nodes.
- **Output Layer:** One node connected to all hidden layer nodes.
- The arrows between nodes represent weights, which are fine-tuned during the training process.

The final ANN model is like a decision-making black box. It takes the features of an email as input and, based on the learned weights and biases, outputs a probability indicating whether the email is likely to be spam.

Example Walkthrough

1. **Input Email:** "Get your free offer now!!! Click here!"
   - **X1 = 5** (5 ad-related words)

- **X2 = 3** (3 exclamation marks)
- **X3 = 2** (2 suspicious links)

2. **Input to ANN:**
   - Neurons take these inputs, calculate weighted sums, apply activation functions, and pass values through the network.
3. **Output:** The output neuron might produce a probability like 0.85.
   - Since 0.85 > 0.5, the network classifies the email as "Spam."

This simplified example illustrates how an ANN functions in a real-world application, such as email classification.