Name: Joselyn Riana Manoj
USN: 1RVU22CSE0071

Ex No: 3 **Date: 21-08-2024**	**Deep Neural Network for Image Classification**

Objective:

The aim of this lab is to design and implement two deep neural networks to classify images in the "Cat vs Non-Cat" dataset. By incorporating multiple layers in the neural network, we seek to enhance the accuracy of the model compared to earlier versions.

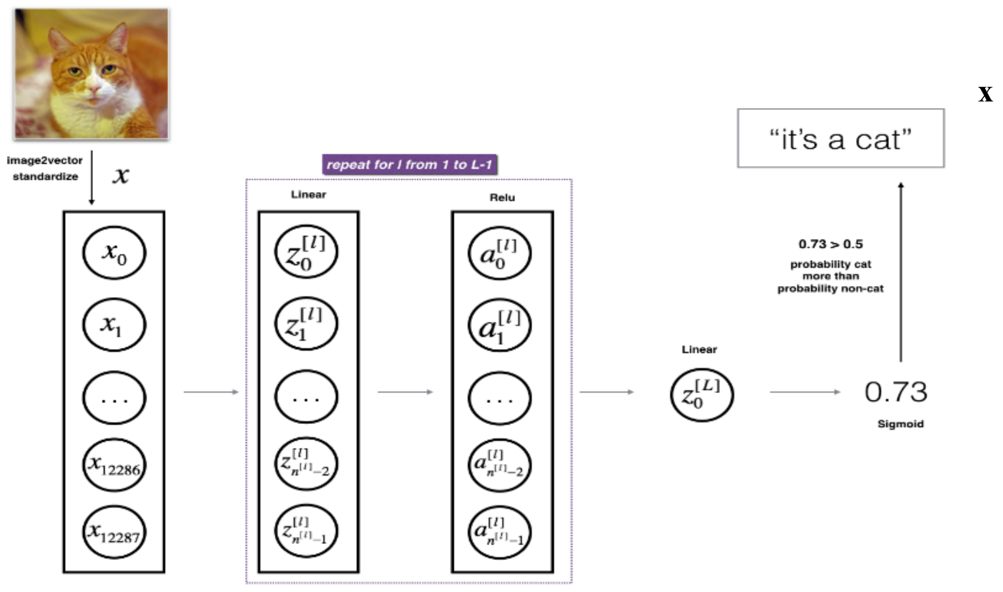
Descriptions:

In this lab, we will develop a deep neural network that classifies images into two categories: cat or non-cat. The dataset comprises images labeled as either cat (1) or non-cat (0), making this a binary classification problem. The task is further complicated by the inclusion of multiple hidden layers within the neural network.

The neural network is structured with both linear and non-linear layers, organized as follows:

- **Input Layer:** Accepts the image data (64x64x3) and flattens it into a single vector.
- **Hidden Layers:** Utilize ReLU activation functions to introduce non-linearity, which enhances the model's ability to detect complex patterns.
- **Output Layer:** Employs a sigmoid activation function to produce a binary output, indicating whether the image is a cat or non-cat.

The model is trained using a cross-entropy loss function, which quantifies the difference between the predicted and actual labels. This loss function guides the model in minimizing errors over multiple training iterations. The parameters of the model will be optimized using backpropagation.



Name: Joselyn Riana Manoj
USN: 1RVU22CSE0071

Build the Model:

- **Initialize Parameters:**
  Define the dimensions of the input layer, hidden layers, and output layer.
- **Forward Propagation:**
  Pass the input through the network, applying linear transformations followed by ReLU and Sigmoid activations.
  Calculate the predicted outputs.
- **Compute Cost:**
  Determine the cross-entropy loss between the predicted and actual labels.
- **Backward Propagation:**
  Use the chain rule to compute gradients of the cost function with respect to the network's parameters.
  Update the parameters using these gradients.
- **Train the Model**:
  Repeat the forward and backward propagation steps over a specified number of epochs until the model converges.

GitHub Link:
https://github.com/joselynrianaaa/DeepLearning_Labs