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

**Objective:**

The objective of this lab is to build and apply two deep neural networks to perform image classification on the "Cat vs Non-Cat" dataset. The goal is to improve upon previous models by increasing the accuracy through the implementation of a deep neural network with multiple layers.

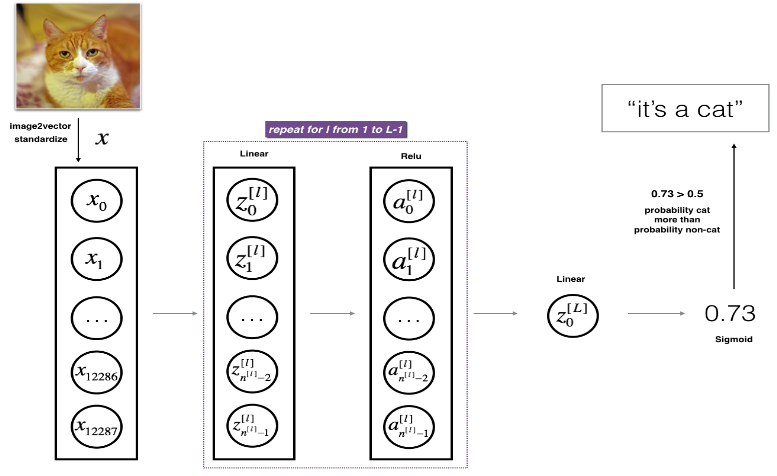
**Descriptions:**

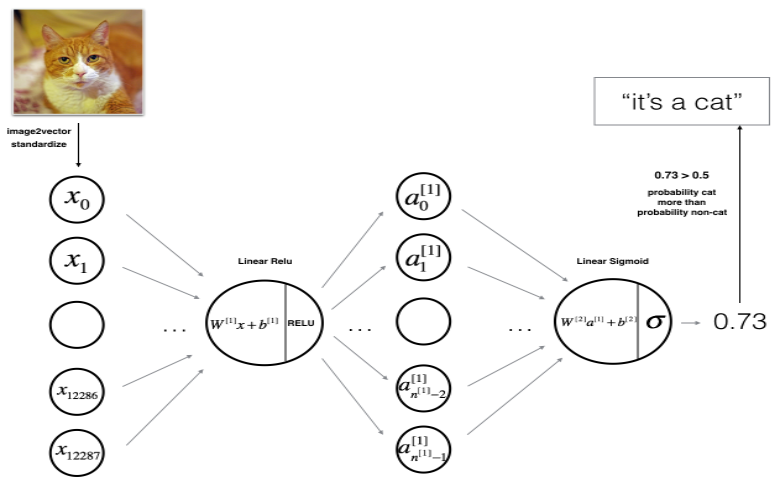
In this lab, we work with a deep neural network model that is designed to classify images into two categories: cat or non-cat. The dataset has images that are labelled either as a cat (1) or non-cat (0). This task is a binary classification problem with the added complexity of multiple hidden layers in the neural network.

The neural network architecture includes both linear and non-linear layers. The layers are structured as follows:

* **Input Layer:** Receives the image data (64x64x3) and flattens it into a vector.
* **Hidden Layers:** Apply ReLU activation functions to introduce non-linearity and improve the network's ability to recognize complex patterns.
* **Output Layer:** Uses a sigmoid function to produce a binary output, indicating whether the image is classified as a cat or non-cat.

The model will be trained using a cross-entropy loss function, which measures the difference between predicted and actual labels, guiding the network to minimize errors over multiple iterations. Backpropagation will be utilized to update the model's parameters and optimize performance.



**x**

**Steps to Build the Model:**

1. **Initialize Parameters:** Define the sizes of the input layer, hidden layers, and output layer.
2. **Forward Propagation:**
   * Pass the input through the layers using linear transformations followed by ReLU and Sigmoid activations.
   * Compute the predicted outputs.
3. **Compute Cost:** Calculate the cross-entropy loss between predicted and actual labels.
4. **Backward Propagation:**
   * Use the chain rule to compute gradients of the cost function concerning the network parameters.
   * Update the parameters based on the gradients.
5. **Train the Model:** Iterate through the forward and backward propagation steps for a set number of epochs until convergence is achieved.

**GitHub Link:**

<https://github.com/abraaaar/RVU_BtechHons/tree/main/Deep%20Learning/Lab%203.2>