|  |  |
| --- | --- |
| **Ex No: 4**  **Date: 20-08-2024** | **Deep Neural Network for Image Classification: Application** |

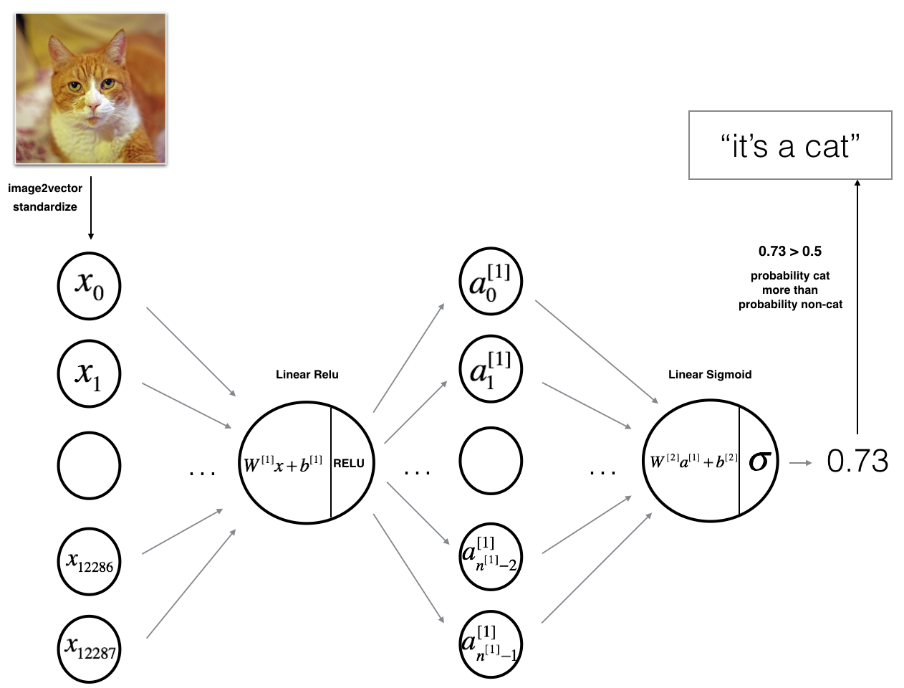
**Objective:**

To develop a deep neural network for the classification of images into cat and non-cat categories. By utilizing a labeled dataset and leveraging deep learning techniques, the project aims to achieve higher accuracy in image classification compared to previous logistic regression models.

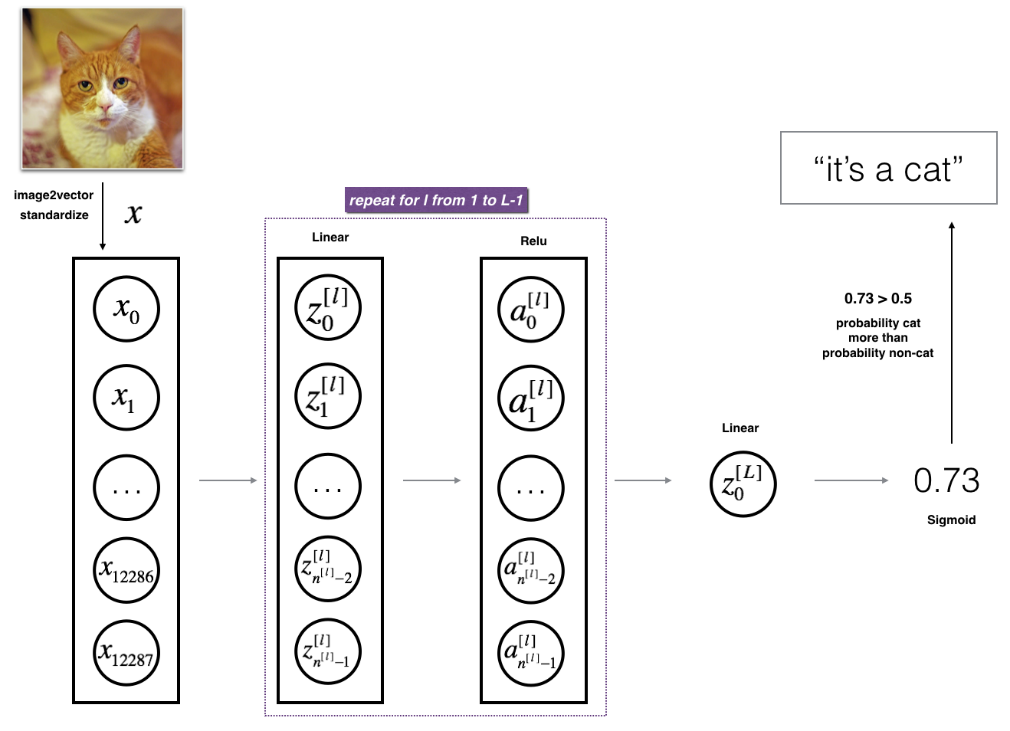
**Descriptions:**

We develop a deep neural network for classifying images as either cat or non-cat. The input to the model consists of images of size (64, 64, 3), which are flattened into a vector of size 12,288. This vector is passed through multiple layers of the neural network. In each layer, the vector is multiplied by a weight matrix, and an intercept is added, forming a linear unit. The ReLU activation function is then applied to the linear unit, introducing non-linearity into the model. This process is repeated across multiple layers, allowing the network to learn complex features of the images. Finally, the output of the last layer is passed through a sigmoid function, which predicts whether the image is of a cat or not. If the output is greater than 0.5, the image is classified as a cat. This deep learning approach is designed to improve upon previous models, such as logistic regression, by better capturing the intricate patterns in image data.

**Model architecture (2-Layer NN):**

****

**Model architecture (L-Layer NN):**

****

**Project Workflow:**

1. Data Preparation:
   * Load and preprocess the "data.h5" dataset containing labeled cat and non-cat images.
   * Flatten each image from (64, 64, 3) to a vector of size (12,288, 1).
2. Model Design:
   * 2-Layer Neural Network: Implement a simple network with one hidden layer (ReLU) and an output layer (Sigmoid).
   * L-Layer Deep Neural Network: Design a deeper network with multiple ReLU hidden layers and a Sigmoid output layer.
3. Training the Model:
   * Forward Propagation: Calculate activations for each layer using ReLU for hidden layers and Sigmoid for the output.
   * Cost Computation: Compute the cost using cross-entropy loss.
   * Backward Propagation: Compute gradients for each parameter.
   * Parameter Updates: Use gradient descent to update weights and biases.
4. Model Evaluation:
   * Use trained parameters to predict test labels and compare the accuracy of the 2-layer and L-layer models.
   * Experiment with different values of LLL to evaluate performance.

Model Architecture:

* Input Layer: Flattened image vector of size (12,288, 1).
* Hidden Layers: Multiple layers with ReLU activations (for L-layer model).
* Output Layer: Sigmoid activation for binary classification (cat vs. non-cat)

**GitHub Link:**

**https://github.com/Bhargava-Srinivasan-26/Deep\_learning\_elective/tree/main/Unit%201/Lab%203\_2**