|  |  |
| --- | --- |
| **Ex No: 3.2**  **Date: 21-08-2024** | **Gradient descent implementation** |

**Objective:**

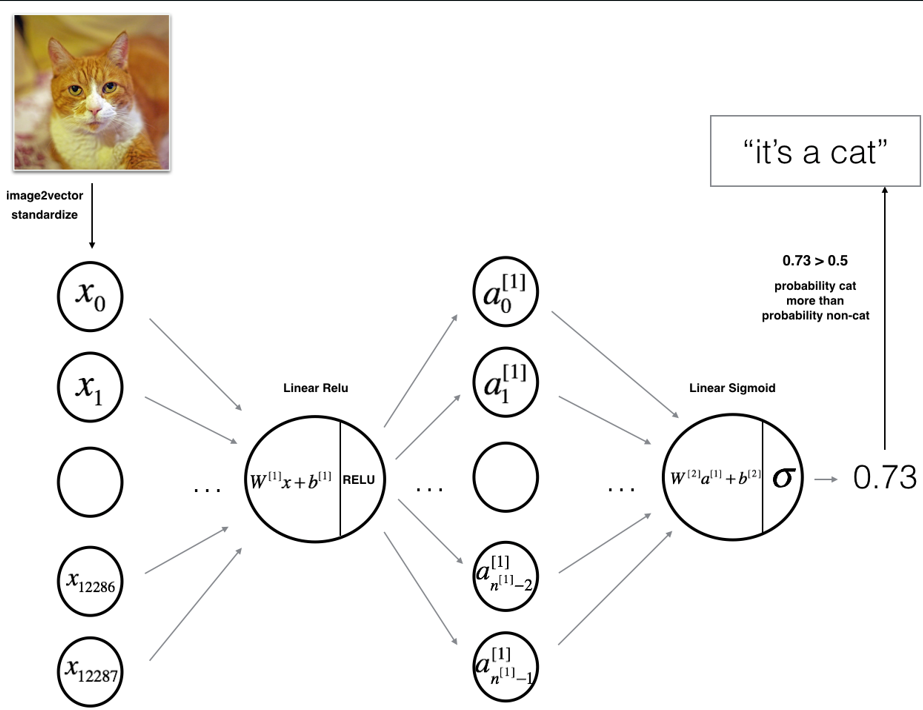
The objective is to create a deep neural network for classifying images as either cat or non-cat. Using a labeled dataset and advanced deep learning methods, this project aims to improve image classification accuracy beyond what was achieved with previous logistic regression models.

**Descriptions:**

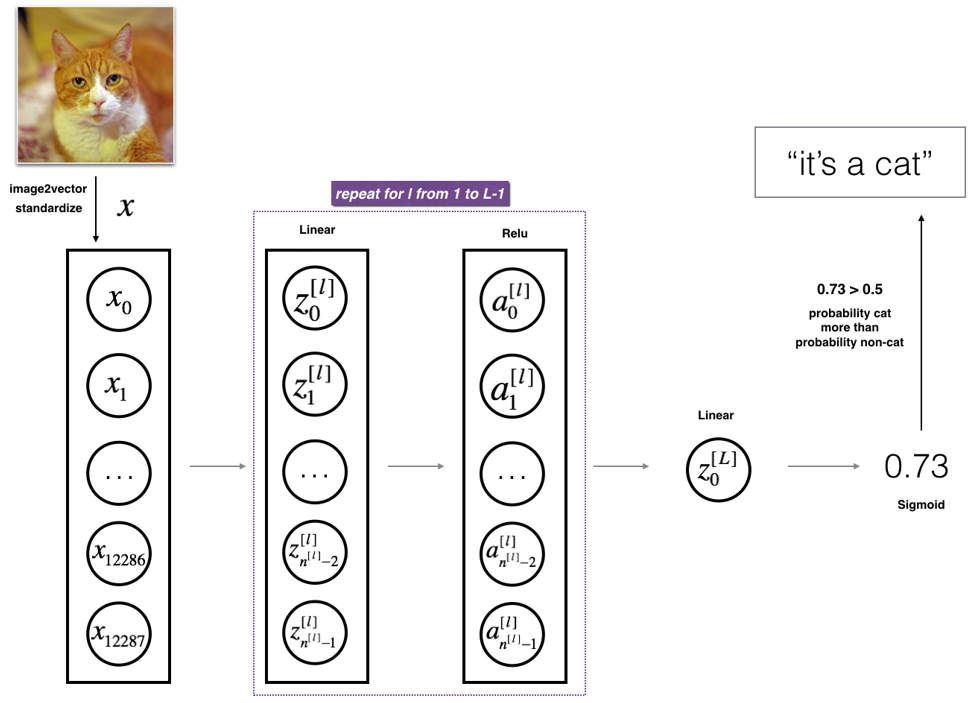
We develop a deep neural network for classifying images as either cat or non-cat. The model takes images of size (64, 64, 3) and flattens them into a vector of size 12,288. This vector is then processed through multiple layers of the network, where in each layer, it undergoes a linear transformation by multiplying it with a weight matrix and adding a bias. The ReLU activation function is applied to introduce non-linearity, enabling the network to capture complex features of the images. This process is repeated through several layers, enhancing the model's ability to learn intricate patterns. The final layer's output is passed through a sigmoid function, which predicts whether the image depicts a cat. If the output exceeds 0.5, the image is classified as a cat. This deep learning approach aims to surpass the accuracy of previous models, such as logistic regression, by more effectively capturing the detailed patterns in image data.

**Model:**

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

****

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

****

**Building the parts of algorithm**

* Data Preparation:

1. Load and preprocess the "data.h5" dataset containing labeled cat and non-cat images.
2. Convert each image from its original size of (64, 64, 3) into a flattened vector of size (12,288, 1).

* Model Design:

1. 2-Layer Neural Network: Construct a basic network with a single hidden layer using ReLU activation, followed by a sigmoid output layer.
2. L-Layer Deep Neural Network: Develop a more complex network with multiple ReLU-activated hidden layers and a sigmoid output layer.

* Model Training:

1. Forward Propagation: Compute activations for each layer, utilizing ReLU for hidden layers and Sigmoid for the final output.
2. Cost Computation: Calculate the cost function using cross-entropy loss.
3. Backward Propagation: Determine gradients for all parameters.
4. Parameter Updates: Adjust weights and biases through gradient descent.

* Model Evaluation:

1. Use the trained model to predict labels on the test set and compare the performance between the 2-layer and L-layer networks.
2. Test different values of LLL to assess model performance.

**GitHub Link:**

**https://github.com/amruthaa-m/DL-Lab1/tree/main/Unit-1/Lab3.2**