|  |  |
| --- | --- |
| **Ex No: 3.2**  **Date: 21/08/2024** | **Deep Neural Network for Image Classification: Application** |

**Objective:**

To build and apply deep neural networks for classifying images into "cat" vs. "non-cat" categories, and to evaluate the performance improvements over previous logistic regression models.

**Description:**

1 - Packages

Begin by importing essential libraries:

2 - Dataset

Use the "Cat vs non-Cat" dataset. This dataset is split into:

- Training set with images labeled as cat (1) or non-cat (0).

- Test set for evaluating model performance.

- Images are reshaped and standardized for input into the neural network.

Steps to prepare the dataset:

- Load Data: Load the dataset containing training and test images and their labels.

- Visualize Data: Display sample images from the dataset and verify their labels.

- Reshape and Standardize: Convert images from 3D (height, width, channels) to 2D vectors. Standardize pixel values to be between 0 and 1.

3 - Architecture of Your Model

Build two types of neural networks:

- 3.1 - 2-Layer Neural Network

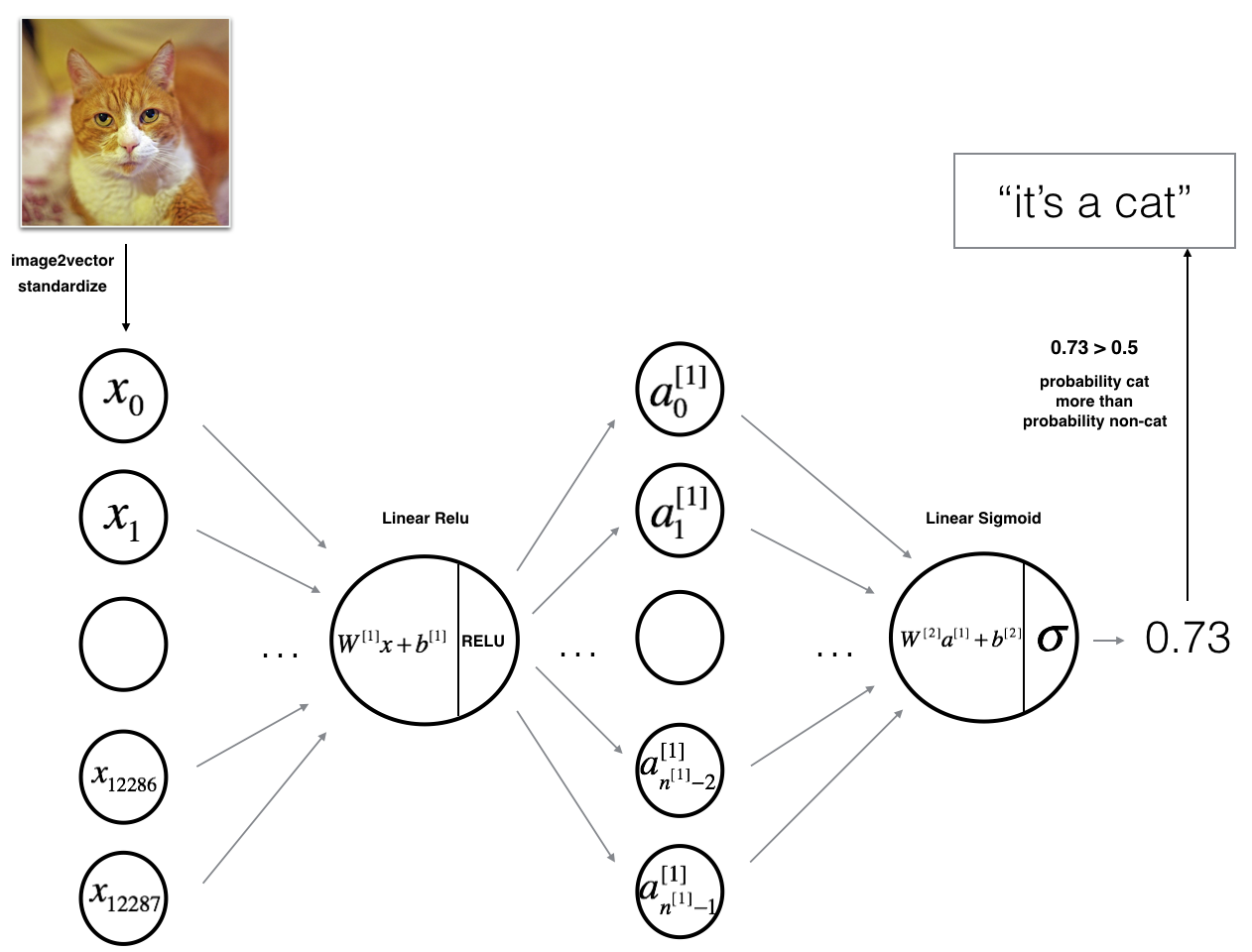
- Structure: Input layer -> Linear transformation -> ReLU activation -> Linear transformation -> Sigmoid activation -> Output layer.

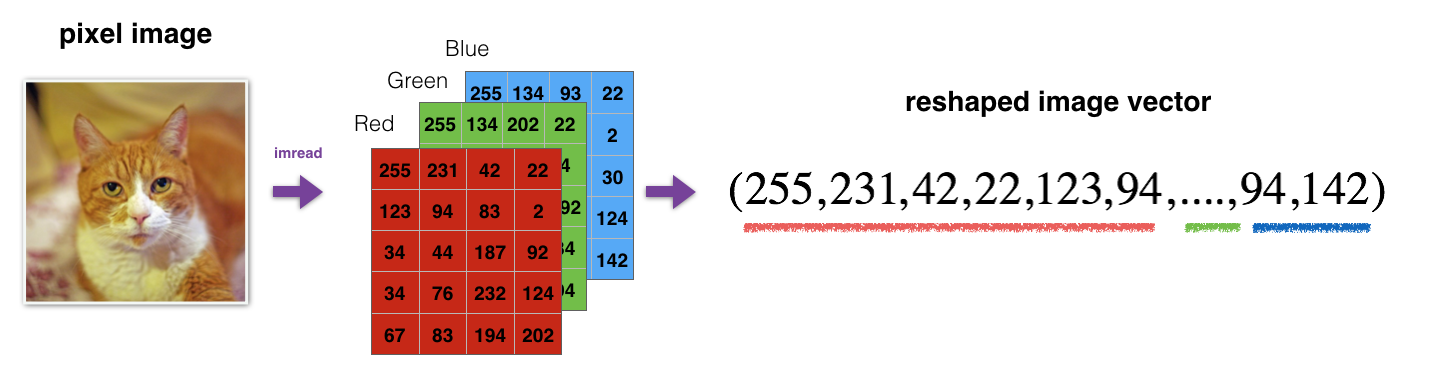
- Components:

- Input: Flattened image vector.

- Hidden Layer: Linear transformation followed by ReLU activation.

- Output Layer: Linear transformation followed by Sigmoid activation.





- Process:

- Initialize parameters.

- Implement forward propagation: Compute activations using ReLU and Sigmoid.

- Calculate cost using the output of forward propagation.

- Perform backward propagation to compute gradients.

- Update parameters using gradient descent.

- Train the model, monitor cost, and evaluate performance.

- 3.2 - L-Layer Deep Neural Network

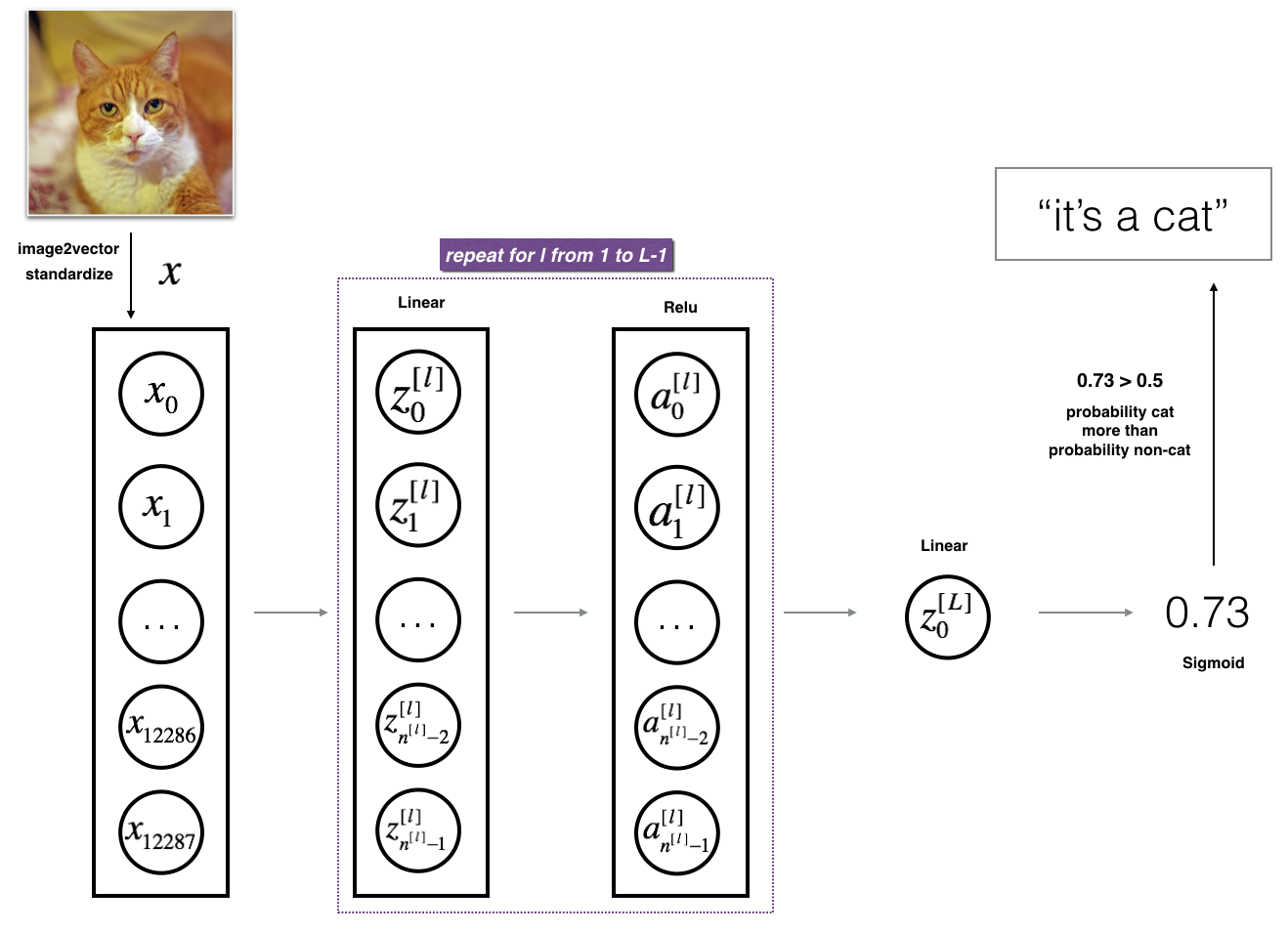
- Structure: Multiple layers of Linear -> ReLU activations, followed by a final Linear -> Sigmoid activation layer.

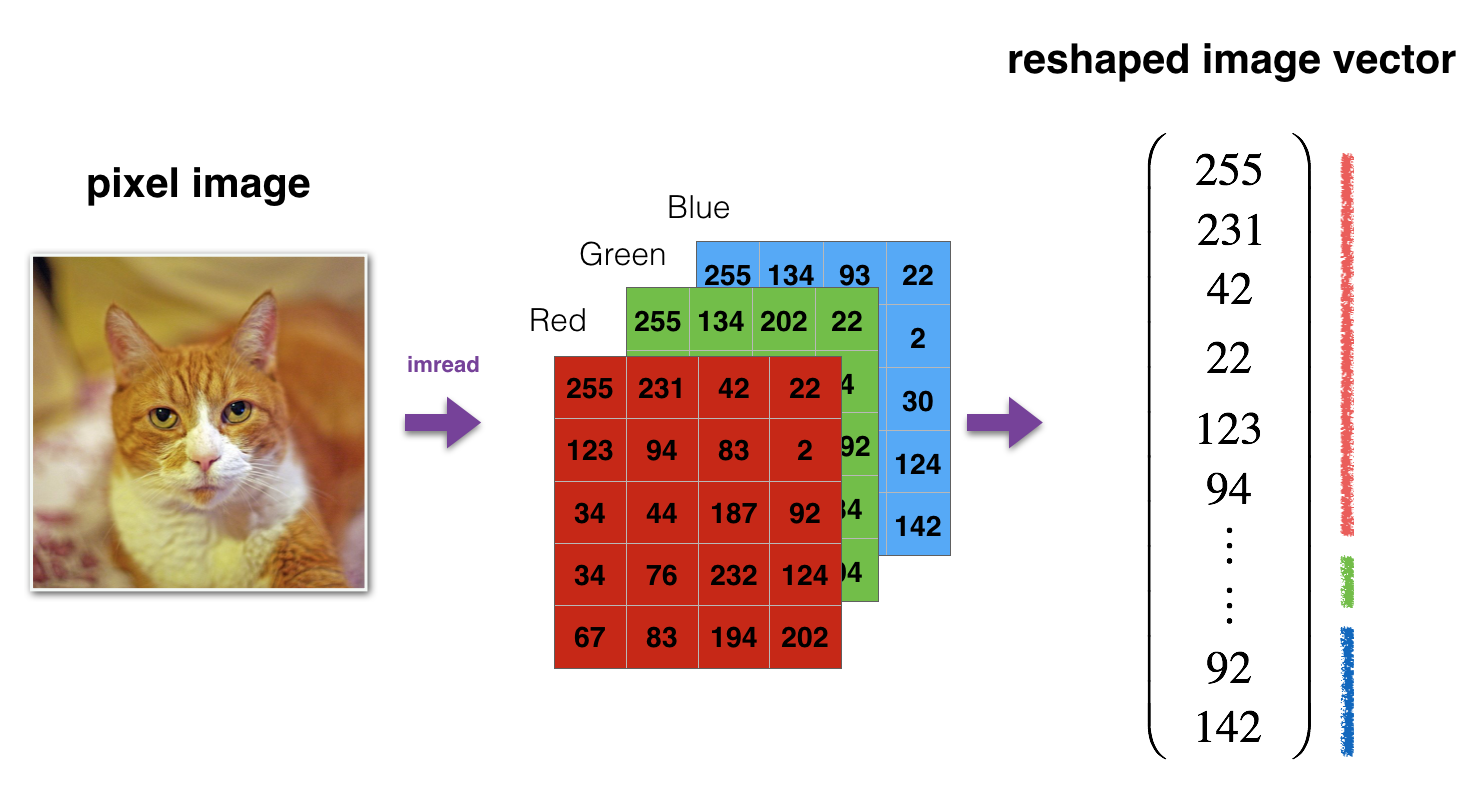
- Components:

- Input: Flattened image vector.

- Hidden Layers: Sequential Linear -> ReLU transformations.

- Output Layer: Final Linear -> Sigmoid activation.





- Process:

- Initialize parameters for all layers.

- Perform forward propagation through all layers.

- Compute cost based on the output.

- Backward propagation through all layers to compute gradients.

- Update parameters using gradient descent.

- Train the model, monitor cost, and evaluate performance.

4 - Two-Layer Neural Network

Implement the 2-layer neural network as outlined, using the provided helper functions for initialization, forward propagation, cost computation, backward propagation, and parameter updating.

5 - L-Layer Neural Network

Implement the L-layer deep neural network, using the provided functions for initializing parameters, forward propagation, cost computation, backward propagation, and parameter updating.

6 - Results Analysis

- Evaluate Models: Compare the performance of the 2-layer and L-layer models. The expectation is that the L-layer model will show improved accuracy over the 2-layer model and logistic regression.

- Analyze Misclassifications: Examine incorrectly classified images to understand potential weaknesses in the model.

7 - Testing with Personal Images (Optional)

- Add a personal image to the dataset.

- Run the model to classify the image and check the result (1 for cat, 0 for non-cat).

The exercise involves implementing and evaluating deep neural networks for image classification. By using a deep neural network instead of logistic regression, one expects to achieve higher classification accuracy and gain experience in training and evaluating more complex models.

**GitHub Link:** https://github.com/dyuthiramesh/Deep\_Learning\_Elective/blob/main/Sem5/Lab3\_2/DNN\_Application\_Distri.ipynb