| **Ex No: 3**  **Date: 21-08-24** | **Deep Neural Network for Image Classification: Application** |
| --- | --- |

**Objective:**

To implement a 2-class classification neural network with A 2-layer neural network and An L-layer deep neural network its performance to a logistic regression model.

**Descriptions:**

Binary classification is the task of classifying elements of a given set into two groups. Logistic regression is an algorithm for binary classification. We have an input image x and the output y is a label to recognize the image. 1 means cat is on an image, 0 means that a non-cat object is in an image. We are building 2 different models

1. A 2-layer neural network

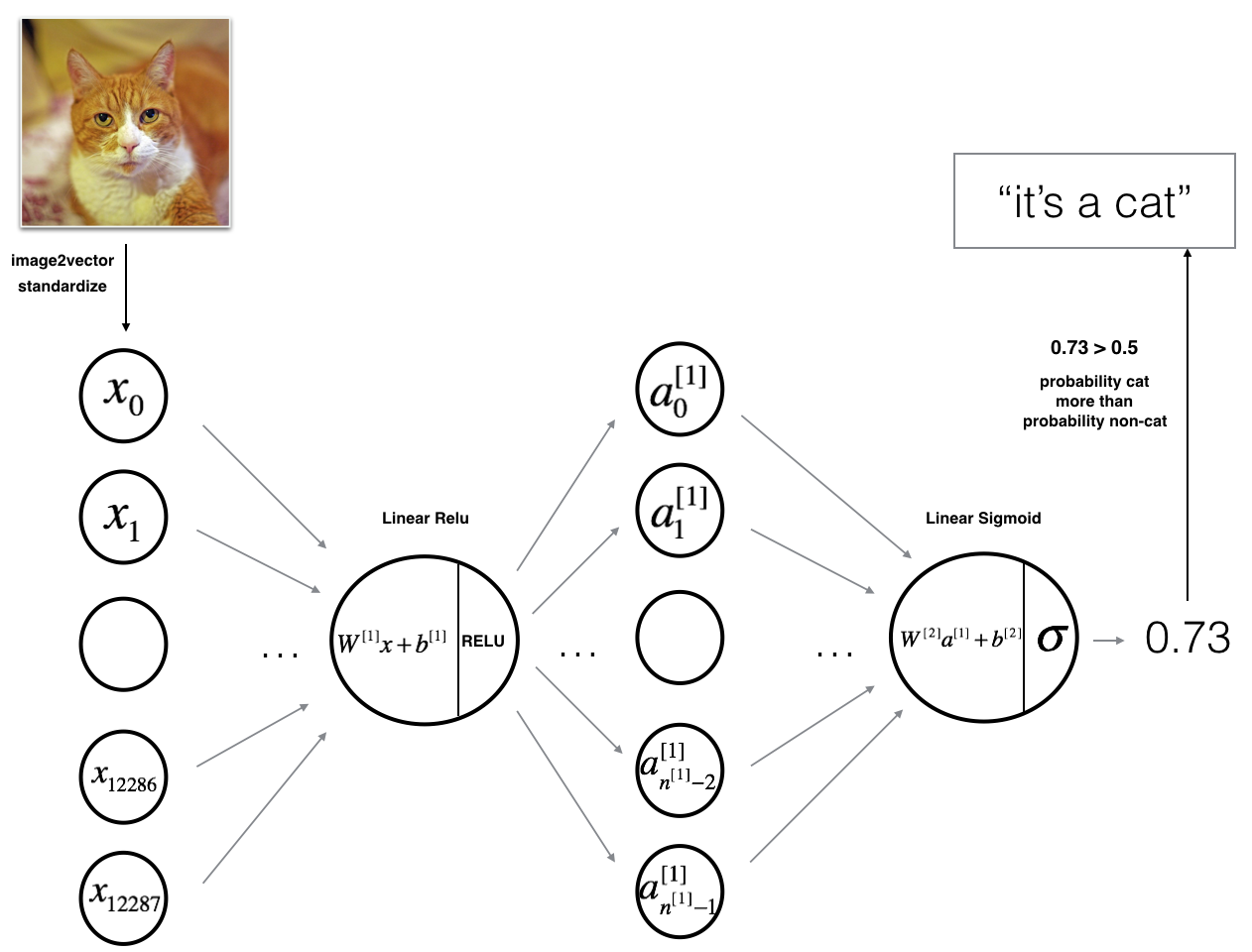
2. An L-layer deep neural network

A 2-layer DNN, as well as deeper neural networks, introduce non-linearity into the model by stacking layers of neurons with non-linear activation functions (e.g., ReLU). This enables the model to capture more complex patterns in the data, allowing for a more flexible decision boundary that can adapt to the intricacies of image data.

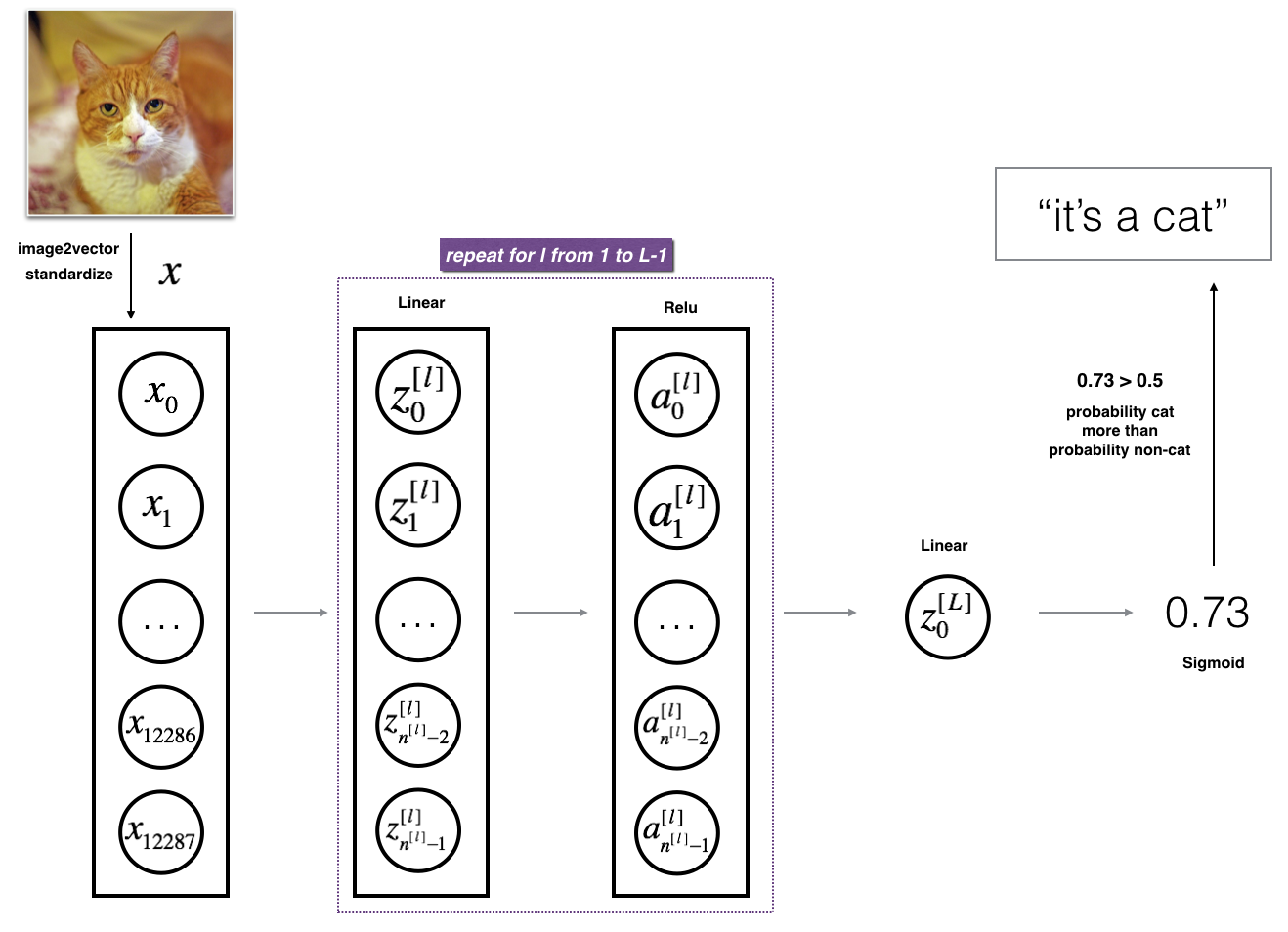
By incorporating hidden layers, the DNN can transform the input data through multiple stages, extracting more abstract features at each layer. This hierarchical feature extraction is crucial in image classification, where higher-level features such as edges, textures, and shapes can be used to better distinguish between classes. The use of multiple layers significantly improves the model's capacity to learn from data, leading to better accuracy on both training and test datasets.



**Model:** A 2-layer neural network

https://piehqfpkotypaenhpraenx.coursera-apps.org/notebooks/Week%202/Logistic%20Regression%20as%20a%20Neural%20Network/images/LogReg_kiank.pnghttps://piehqfpkotypaenhpraenx.coursera-apps.org/notebooks/Week%202/Logistic%20Regression%20as%20a%20Neural%20Network/images/LogReg_kiank.png

An L-layer deep neural network



Mathematically:

y^​=f(Wx+b)

Where:

* x is the input (reshaped image),
* W is the weight matrix,
* b is the bias vector,
* f is the activation function (e.g., sigmoid, ReLU),
* y^\hat{y}y^​ is the predicted output (in this case, whether the image is a cat or non-cat).

**Building the parts of algorithm**

### 1. Packages

* Import necessary libraries such as NumPy, imageio, PIL, Matplotlib, and any deep learning frameworks (e.g., TensorFlow, PyTorch).

### 2. Dataset

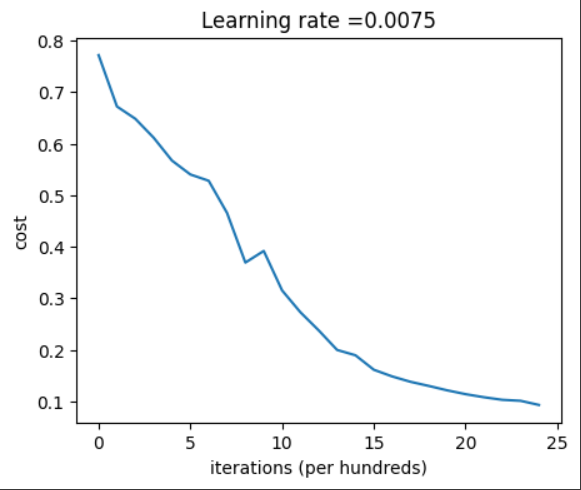
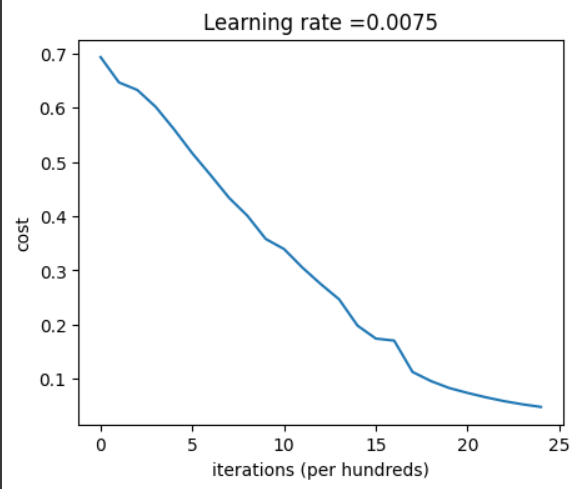
* Load and preprocess the dataset.
* Ensure images are resized, normalized, and reshaped to match the model’s input dimensions.

### 3. Model Architecture

* **A 2-layer Neural Network**:
  + Create or load a 2-layer neural network, with a hidden layer and an output layer for binary classification (e.g., cat vs. non-cat).
  + LINEAR -> RELU -> LINEAR -> SIGMOID.
* **A L-layer Deep Neural Network**:
  + Load or define an L-layer deep neural network with multiple hidden layers, leveraging techniques such as ReLU activation, dropout, etc.
  + \*[LINEAR -> RELU] × (L-1) -> LINEAR -> SIGMOID\*
* **General Methodology**:
  + Define the forward propagation, activation functions, and loss calculation.
  + Implement backpropagation and optimization steps if training from scratch.

### 4. Results Analysis

* Evaluate the model's performance using metrics like accuracy, precision, recall, etc.
* Analyze the results to assess model effectiveness and identify areas for improvement.



**GitHubLink:** [**https://github.com/chandanab1/Deep\_Learning**](https://github.com/chandanab1/Deep_Learning)