USN:1RVU22CSE153
NAME:SHREYAS R

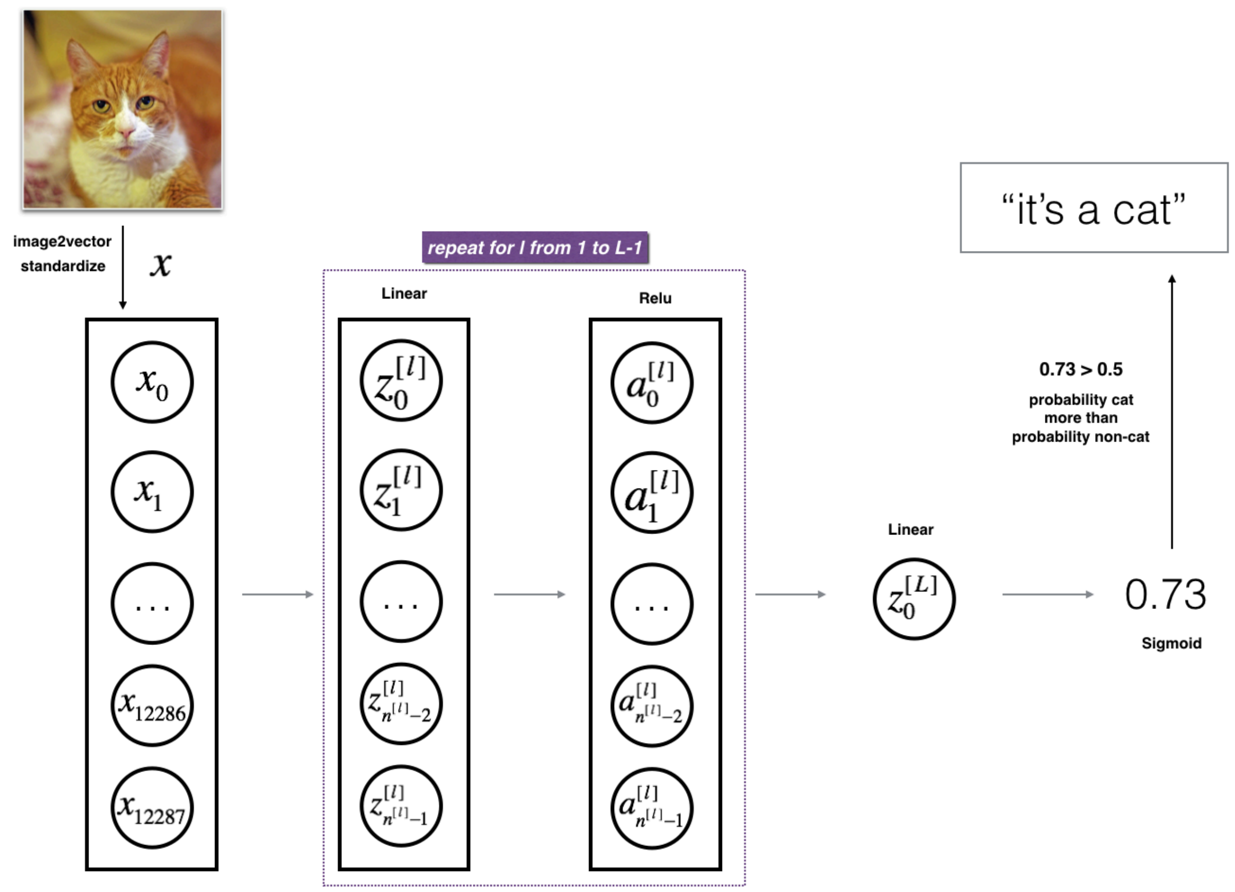
Title: Deep Neural Network for Image Classification

Objective:

To build a Deep Neural Network to classify images as cat and non-cat categories

Description:

In this experiment, we aim to implement a Deep Neural Network (DNN) for binary classification. The model will take an input image X and output a prediction y, where y=1 indicates that the image contains a cat, and y=0 indicates a non-cat object. The DNN model consists of multiple layers with weights initialized randomly. The model undergoes forward propagation to compute the loss and backward propagation to update the weights using gradient descent.



USN:1RVU22CSE153
NAME:SHREYAS R

A Deep Neural Network (DNN) is a supervised learning algorithm used for complex classification tasks, such as distinguishing between cat and non-cat images. Unlike logistic regression, which has no hidden layers, a DNN consists of multiple layers of neurons that learn hierarchical representations of the input data. Each layer applies a linear transformation followed by a non-linear activation function, such as ReLU for hidden layers and sigmoid for the output layer. The initial weights are typically initialized randomly. As the model iterates through the training data, the weights are adjusted based on the gradients of the loss function, enabling the DNN to gradually learn the features necessary to accurately classify the images. Over time, the network fine-tunes these weights to minimize the error, improving its predictive performance with each iteration.

Model:

1. **Initialization**:
   - Initialize weights `W1` and `W2` for the hidden and output layers respectively.
   - Initialize biases `b1` and `b2`.
2. **Forward Propagation**:
   - **Layer 1**: Compute `Z1 = W1*X + b1` and apply the ReLU activation function.
   - **Layer 2**: Compute `Z2 = W2*A1 + b2` and apply the sigmoid activation function to get `A2`.
3. **Cost Function**:
   - Compute the binary cross-entropy loss between the predicted output `A2` and the true label `Y`.
4. **Backward Propagation**:
   - Compute gradients `dW1`, `db1`, `dW2`, `db2` with respect to the cost.
   - Update weights and biases.
5. **Training**:
   - Loop over a number of iterations to optimize the parameters.

Results:

- The model is trained using the training set over 2500 iterations.
- The cost decreases over iterations, indicating successful learning.

Training Accuracy	0.9856459330143539
Testing Accuracy	0.8

USN:1RVU22CSE153
NAME:SHREYAS R

Github Link: **https://github.com/shreyasrajiv327/CS3232-DeepLearning/tree/main/Lab3.2**