| Ex No: 3.1  Date: 20-08-2024 | Building a Deep Neural Network for Image Classification |
| --- | --- |

**Objective:**

To build a deep neural network model with multiple layers to classify images into different categories. This lab focuses on constructing a deep learning model from scratch, understanding the architecture, and training it to achieve high accuracy.

**Descriptions:**

Deep learning models can handle large-scale and complex datasets by learning hierarchical representations of the data. In image classification tasks, deep neural networks (DNNs) are especially powerful due to their ability to capture intricate image patterns across multiple layers.

The goal of this exercise is to build a deep neural network with several layers (including hidden layers) and apply it to classify images into various categories. This involves a step-by-step process of defining the network architecture, initializing the parameters, and training the model using forward and backward propagation techniques.

Deep neural networks consist of:

* **Input Layer:** Takes in the raw pixel data from images.
* **Hidden Layers:** Multiple layers that apply non-linear transformations to capture complex patterns.
* **Output Layer:** Produces the final classification result.

The depth of the network, determined by the number of hidden layers, allows the model to learn more abstract features, which improves classification accuracy for complex tasks like image recognition.

**Model:**

**Steps to Build a Deep Neural Network**

1. **Define the model architecture:**
   * Specify the number of layers and the number of units in each layer.
   * Choose activation functions (e.g., ReLU, sigmoid) for each layer.
2. **Initialize parameters:**
   * Randomly initialize weights for all layers.
   * Set up biases for each layer.
3. **Implement forward propagation:**
   * Pass the input through each layer to compute the output.
   * Apply activation functions after each layer to introduce non-linearity.
4. **Compute the cost:**
   * Use a loss function (e.g., cross-entropy) to measure the difference between the predicted output and the actual labels.
5. **Implement backward propagation:**
   * Calculate gradients of the cost function with respect to the parameters.
   * Use these gradients to update the parameters.
6. **Train the model:**
   * Run the forward and backward propagation iteratively over the training set.
   * Use an optimization algorithm like gradient descent to minimize the cost.
7. **Evaluate the model:**
   * Test the trained model on a validation or test set to assess its performance.

**GitHub Link:**

https://github.com/SudhanvaMS2004/Deep\_Learning\_Fundamentals/blob/main/Building\_Deep\_Neural\_Network\_Distri.ipynb