|  |  |
| --- | --- |
| **Ex No: 2**  **Date: 27th August 2024** | **Implementing a Multi-Class Classification Model using Deep neural network** |

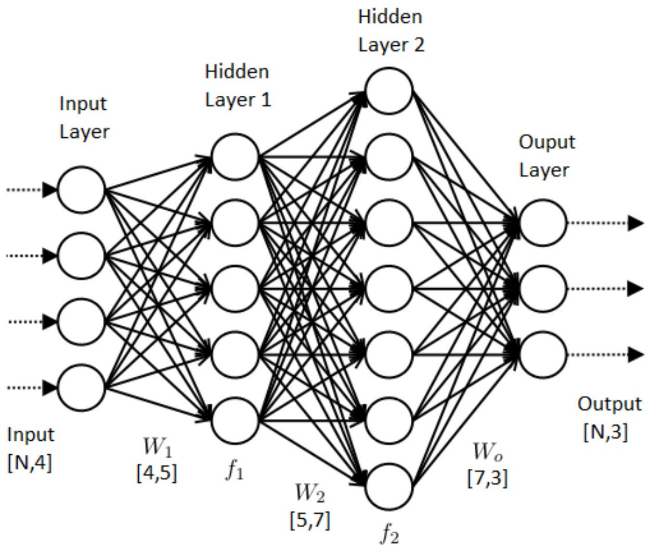
**Objective:** To build and train a deep neural network (DNN) model capable of classifying images of flowers into one of five categories: daisy, dandelion, rose, sunflower, or tulip. This exercise aims to develop skills in designing, implementing, and optimizing multi-class classification models using neural networks.

**Descriptions:**

An L-layer deep neural network is a powerful model architecture designed to capture complex patterns in data through multiple layers of nonlinear transformations. Each layer in the network consists of neurons that apply linear transformations followed by activation functions, allowing the network to learn and represent intricate relationships within the input data. The depth of the network, indicated by the number of layers ***L****, enables* it to model hierarchical features, where lower layers capture basic features (like edges in an image), and higher layers combine these features into more abstract representations (such as shapes or objects).

In a deep neural network, forward propagation involves passing the input through each layer sequentially, with each layer's output serving as the input for the next. The final layer typically uses the softmax activation function to produce a probability distribution over the possible classes, which is particularly useful for multi-class classification tasks. During training, the model parameters (weights and biases) are optimized using backpropagation and gradient descent, ensuring that the network learns to make accurate predictions by minimizing a chosen loss function, such as cross-entropy for classification problems. This L-layer architecture is highly adaptable and forms the basis of many state-of-the-art models in various machine learning applications.

**Model:**



Sample example of L-Layer Deep Neural Network for Multi-class classification

Image Source: [Click here](https://medium.com/@srijaneogi31/exploring-multi-class-classification-using-deep-learning-cd3134290887)

**Building the parts of the algorithm**

Here are the steps involved in building each part of the algorithm:

**1. Load and Preprocess the Data**

* Load the training and test data from the provided files (Tr.h5 and Te.h5).
* Flatten the image data and normalize pixel values to be between 0 and 1.
* Convert labels to one-hot encoding to be used with the softmax output.

**2. Initialize Parameters**

* Define the number of layers and the number of units in each layer.
* Initialize weights using a small random value or specific initialization methods like Xavier or He initialization.
* Initialize biases to zero.

**3. Forward Propagation**

* Implement a function for forward propagation through the network.
* Apply activation functions: ReLU for hidden layers and softmax for the output layer.
* Compute the output probabilities for each class.

**4. Compute Cost**

* Implement the cross-entropy loss function to compute the cost.
* This function takes the predicted probabilities and actual labels as inputs.

**5. Backward Propagation**

* Calculate the gradient of the loss function with respect to each parameter (weights and biases) using backpropagation.
* Implement derivative calculations for each activation function used (ReLU, softmax).

**6. Update Parameters**

* Implement gradient descent or another optimization algorithm to update the parameters.
* Use the calculated gradients to adjust the weights and biases.

**7. Train the Model**

* Loop through multiple epochs, performing forward propagation, cost computation, backward propagation, and parameter update in each epoch.
* Track the cost function's value to monitor the model's learning process.

**8. Evaluate the Model**

* Test the model using the test dataset to compute its accuracy.
* Compare the predicted labels with the actual labels to assess performance.
* Analyze errors to understand potential improvements.

**GitHub Link:** [**https://github.com/tulasigr/DeepLearning**](https://github.com/tulasigr/DeepLearning)