|  |  |
| --- | --- |
| **Ex No: 2**  **Date: 15-08-2024** | **Implementing a Multi-Class Classification Model using Deep neural network** |

**Objective:**

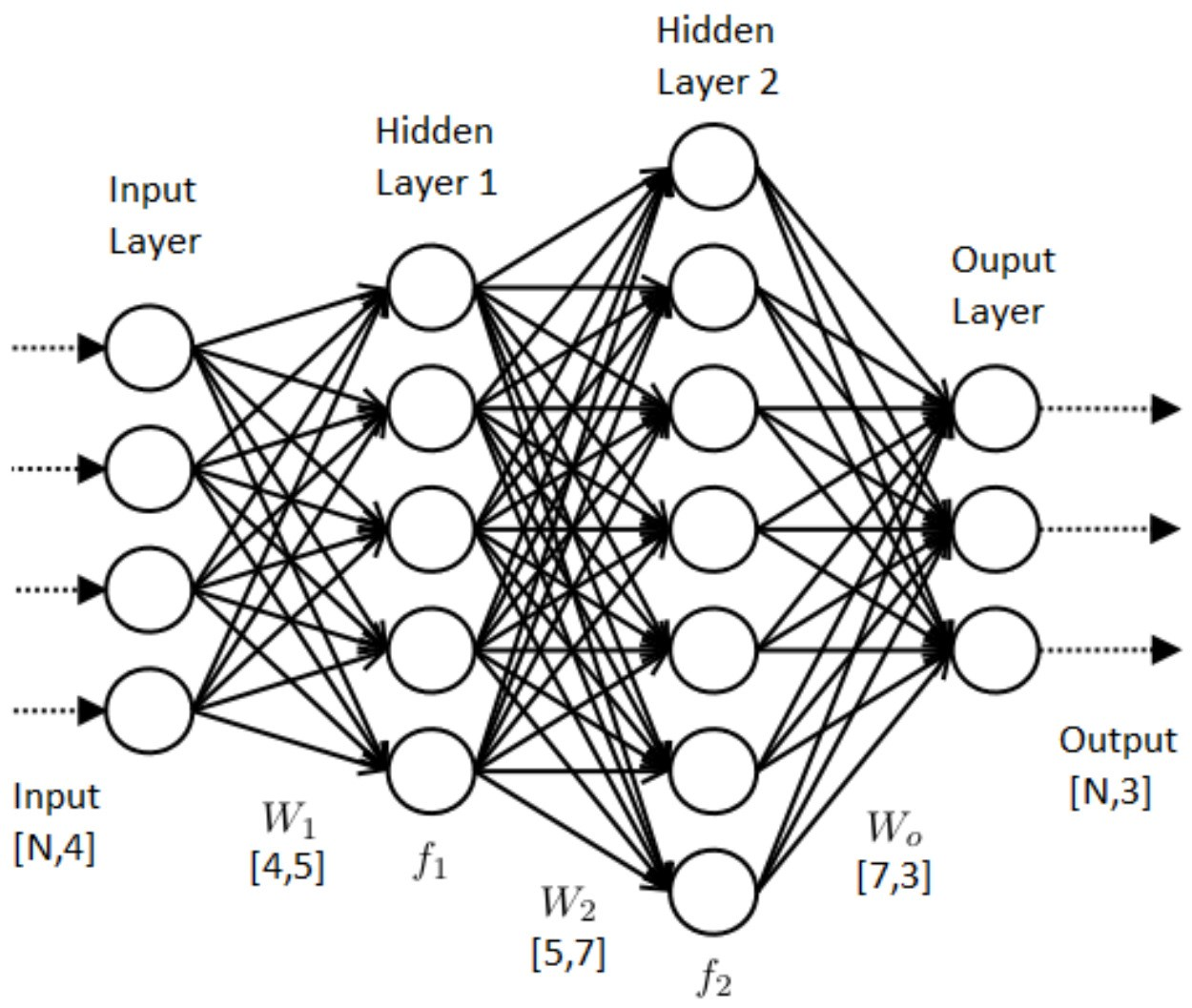
The goal of this project is to develop a deep neural network (DNN) to classify images of flowers into multiple categories. The model will distinguish between five flower types: Daisy, Dandelion, Rose, Sunflower, and Tulip. By training and testing the model on a provided dataset, we aim to achieve high classification accuracy. This project will help in understanding the application of deep learning in image classification tasks.

**Descriptions:**

The dataset consists of images with corresponding labels for training and testing the model. Images are preprocessed by flattening and normalizing pixel values, and labels are converted into a one-hot encoded format. The model will use a deep neural network architecture to handle the multi-class classification problem effectively. The performance of the model will be evaluated based on its accuracy on the test dataset.

In this project, we utilize a dataset containing images of various flowers, each associated with a label indicating its category. The data preprocessing involves flattening each image to a vector and normalizing pixel values to a range between 0 and 1, which helps in speeding up the training process. Labels are converted into a one-hot encoded format to facilitate multi-class classification. The deep neural network is designed with multiple layers, including hidden layers with ReLU activation and an output layer with Softmax activation. This architecture allows the model to learn complex patterns and make probabilistic predictions for each flower category. The performance of the model is evaluated based on its accuracy in classifying images correctly on a separate test dataset, providing insights into the model's generalization ability and effectiveness in handling multi-class classification problems.

**Model:**



The model consists of multiple layers: an input layer, two hidden layers with ReLU activation, and an output layer with Softmax activation. The architecture includes 64 and 32 units in the hidden layers. The Softmax function is used in the output layer to handle multi-class classification by computing probabilities for each class. The model parameters are optimized using gradient descent to minimize the cross-entropy loss function.

**Building the parts of algorithm**

1. **Define the Model Structure:**

* Specify the input layer (flattened image size).
* Set up hidden layers with ReLU activation functions.
* Configure the output layer with Softmax activation for classification.

1. **Initialize the Model's Parameters:**

* Initialize weights and biases for each layer.
* Use small random values for weights and zeros for biases.

1. **Loop:**

* **Calculate Current Loss (Forward Propagation):**
  + Perform forward propagation to compute the predicted probabilities and loss using the cross-entropy function.
* **Calculate Current Gradient (Backward Propagation):**
  + Compute gradients of weights and biases using backward propagation to update the model parameters.
* **Update Parameters (Gradient Descent):**
  + Adjust weights and biases based on computed gradients and learning rate to minimize the loss function.

**GitHub Link:**