STEP BY STEP PROCESS OF WORK EXECUTION

1. Setup:

Set up your development environment with Python and the necessary libraries, including TensorFlow and Keras.

1. Download Dataset:

Download a labeled dataset of flower images from reliable sources or datasets specifically curated for flower recognition tasks. Ensure the dataset includes images of various flower species.We had taken data set of flowers with 5 species.

1. Explore The Data Set

Like , you must explore the data set first before further functioning . check the length of data set , print random images form data set by indexing.

1. Load Data Using Keras:

Use Keras, a high-level deep learning library, to load the dataset into your Python environment. Keras provides convenient functions for loading common datasets, such as CIFAR-10 or ImageNet, but we used custom functions to load your downloaded dataset.

1. Create Dataset:

Create separate arrays to store the images and their corresponding labels from the loaded dataset. This will allow you to manipulate and preprocess the data easily. like batch size = 32, img\_height=180, img\_width=180.

1. Visualize Data:

Visualize the loaded flower images to gain an understanding of the dataset. Use libraries like Matplotlib or Seaborn to plot sample images along with their corresponding labels to ensure the data is correctly labeled.

1. Configure Data for Performance:

Ensure that the dataset is balanced, with an equal number of images for each flower species. If there is an imbalance, consider techniques like oversampling or undersampling to address it.

1. Standardize Data:

Standardize the pixel values of the images by dividing them by 255. This step ensures that the pixel values are within the range of 0 to 1, which aids in convergence during model training.

1. Create a Basic Keras Model:

Design a basic CNN model using Keras. Start with convolutional layers followed by activation functions like ReLU, max pooling layers for downsampling, and additional convolutional and pooling layers to learn hierarchical features.

1. Compile Model:

Compile the model using an appropriate optimizer, such as Adam, and a suitable loss function, such as categorical cross-entropy, for multi-class classification. This step prepares the model for training.

1. Model Summary:

Print the summary of the model architecture, which provides an overview of the model's layers, parameters, and output shapes.

1. Train Model:

Split the dataset into training and validation sets. Use the training set to train the model and the validation set to monitor the model's performance during training.

Train the model using the compiled model, training dataset, a specified number of epochs, and appropriate batch size.

1. Visualize Training Results:

Plot and analyze the training and validation accuracy and loss curves over epochs. This visualization helps to monitor the model's performance and detect any signs of overfitting or underfitting.

1. Overfitting Found:

Overfitting is observed (when the validation loss starts to increase while the training loss continues to decrease), take steps to address it. Apply techniques like data augmentation and dropout regularization.

1. Data Augmentation:

Implement data augmentation techniques, such as random rotations, translations, flips, and zooming, to artificially increase the dataset's size and diversity. This improves the model's generalization capability and reduces overfitting.

1. Dropout:

Introduce dropout regularization by adding Dropout layers to the model. Dropout randomly deactivates neurons during training, preventing over-reliance on specific features and improving generalization.

1. Compile and Train the Model:

Recompile the model with the same optimizer and loss function.

Train the model again using the augmented data and the updated model architecture with dropout layers.

1. Visualize Training Results Again:

Plot and analyze the training and validation accuracy and loss curves after implementing data augmentation and dropout. Check if the overfitting has been mitigated and if the model's performance has improved.

1. Predict New Data:

Use the trained model to make predictions on new, unseen flower images. Evaluate the model's accuracy and performance on this separate test dataset to assess its ability to generalize to unseen data.