|  |  |
| --- | --- |
| **Ex No: 5**  **Date: 04-09-2024** | **Transfer Learning** |

**Objective:**

The objective of this lab is to build a flower classification model using the MobileNet V2 pre-trained model from TensorFlow Hub. The model will classify different types of flowers (daisy, dandelion, roses, sunflowers, and tulips) using transfer learning. We will preprocess the dataset, train the model, and evaluate its performance.

**Descriptions:**

The code demonstrates how to use the MobileNet V2 model, pre-trained on ImageNet, for flower classification. The model is first loaded using TensorFlow Hub and frozen so that its weights are not updated during training. The dataset is prepared by downloading a set of flower images, resizing them to the required dimensions, and normalizing the pixel values. A new model is then created using the pre-trained model as a feature extractor, followed by a dense layer to classify the images into the five flower categories. The model is trained, evaluated, and then used to make predictions on unseen images.

**Steps to Build the Model:**

**Install Necessary Libraries**: Install TensorFlow, TensorFlow Hub, and OpenCV.

**Import Libraries**: Import all required libraries, such as TensorFlow, TensorFlow Hub, OpenCV, PIL, NumPy, etc.

**Define Image Shape**: Specify the shape of the images to be used.

**Load Pre-Trained Model**: Use a pre-trained MobileNet V2 model from TensorFlow Hub for feature extraction.

**Load and Preprocess Images**: Load images from the dataset, resize, normalize, and prepare them for input to the model.

**Create a Model**: Define a Sequential model using the pre-trained MobileNet V2 as a feature extractor and add a dense output layer for classification.

**Compile the Model**: Compile the model with an appropriate loss function and optimizer.

**Train the Model**: Fit the model using the training dataset.

**Evaluate the Model**: Evaluate the model's performance on the test dataset.

**Make Predictions**: Use the trained model to predict the class of new images.

feature\_extractor\_model = "https://tfhub.dev/google/tf2-preview/mobilenet\_v2/feature\_vector/4"

pretrained\_model\_without\_top\_layer = hub.KerasLayer(

    feature\_extractor\_model, input\_shape=(224, 224, 3), trainable=False)

num\_of\_flowers = 5

model = tf\_keras.Sequential([

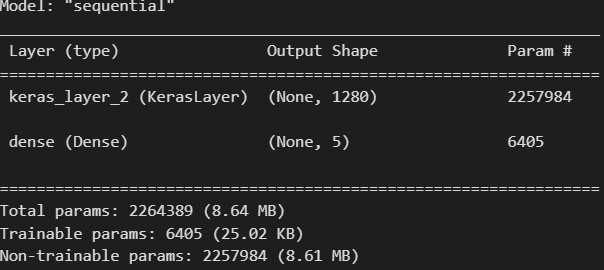
  pretrained\_model\_without\_top\_layer,

  tf\_keras.layers.Dense(num\_of\_flowers)

])

Selecting a MobileNetV2 pre-trained model from TensorFlow Hub and wrapping it as a Keras layer

The original model is set to non-trainable to prevent updates to its weights during training. We are only training the Dense Layer at the end here.

The model itself is :  


We defined an image pre processing function as given below :

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

def preprocess\_image(image\_path):

    img = load\_img(image\_path, target\_size=(224, 224))

    img\_array = img\_to\_array(img)

    img\_array = np.expand\_dims(img\_array, axis=0)  # Add batch dimension

    img\_array /= 255.0  # Normalize to [0, 1] range

    return img\_array

we load the image and convert it into a numpy array using inbuilt functions.

Next we add a new dimension so that it’s the correct size before passing it into the model

Then we normalize the values by dividing by the maximum possible value(255.0)

Next, we compile the model with our loss function set as categorical cross entropy, and Adam optimizer.

model.compile(

  optimizer="adam",

  loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

  metrics=['acc'])

Then we train the model using the fit() method:

model.fit(X\_train\_scaled, y\_train, epochs=5)

on running

model.evaluate(X\_test\_scaled,y\_test)

we find the accuracy to be 86.71% on the test set