**Ex No: 5 TRANSFER LEARNING WITH CNN AND VISUALIZATION**

**Aim:**

To build a convolutional neural network with transfer learning and perform visualization

**Procedure:**

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow. 4. Compile and fit the model.
4. Perform prediction with the test dataset.
5. Calculate performance metrics. **CODE:**

import tensorflow\_datasets as tfds from keras.utils import to\_categorical import tensorflow as tf

*## Loading images and labels*

(train\_ds, train\_labels), (test\_ds, test\_labels) = tfds.load(

"tf\_flowers", split=["train[:70%]", "train[:30%]"], *## Train test split* batch\_size=-1, as\_supervised=True, *# Include labels*

)

*## Resizing images* train\_ds = tf.image.resize(train\_ds, (150, 150)) test\_ds = tf.image.resize(test\_ds, (150, 150))

*## Transforming labels to correct format* train\_labels = to\_categorical(train\_labels, num\_classes=5) test\_labels = to\_categorical(test\_labels, num\_classes=5) from tensorflow.keras.applications.vgg16 import VGG16 from tensorflow.keras.applications.vgg16 import preprocess\_input *## Loading VGG16 model* base\_model = VGG16(weights="imagenet", include\_top=False, input\_shape=train\_ds[0].shape) base\_model.trainable = False *## Not trainable weights*

*## Preprocessing input* train\_ds = preprocess\_input(train\_ds) test\_ds = preprocess\_input(test\_ds) from tensorflow.keras.callbacks import EarlyStopping model**.**compile(

optimizer**=**'adam', loss**=**'categorical\_crossentropy', metrics**=**['accuracy']

)

*# Set up early stopping* es **=** EarlyStopping(monitor**=**'val\_accuracy', mode**=**'max', patience**=**5, restore\_best\_weights**=**True)

*# Train model and store history*

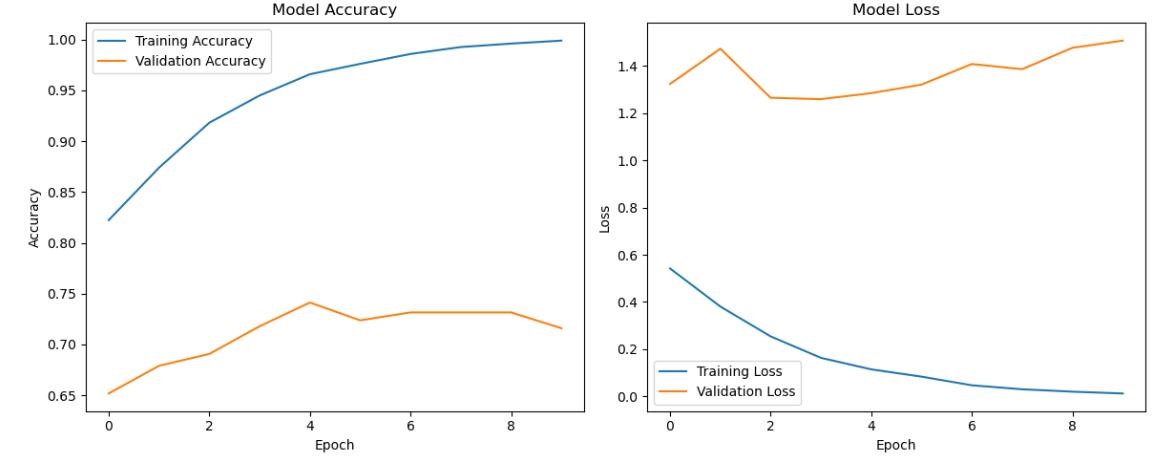
history **=** model**.**fit(train\_ds, train\_labels, epochs**=**20, validation\_split**=**0.2, batch\_size**=**32, callbacks**=**[es])

import matplotlib.pyplot as plt *# Plot accuracy and loss* plt.figure(figsize=(12, 5))

*# Accuracy plot* plt.subplot(1, 2, 1) plt.plot(history.history['accuracy'], label='Training Accuracy') plt.plot(history.history['val\_accuracy'], label='Validation Accuracy') plt.xlabel('Epoch') plt.ylabel('Accuracy') plt.legend(loc='best') plt.title('Model Accuracy')

*# Loss plot* plt.subplot(1, 2, 2) plt.plot(history.history['loss'], label='Training Loss') plt.plot(history.history['val\_loss'], label='Validation Loss') plt.xlabel('Epoch') plt.ylabel('Loss') plt.legend(loc='best') plt.title('Model Loss')

plt.tight\_layout() plt.show() **Output:**



**Result:**

Thus transfer learning with cnn was implemented successfully.