ROLLNO:210701120

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.

5. Perform prediction with the test dataset.

6. 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')

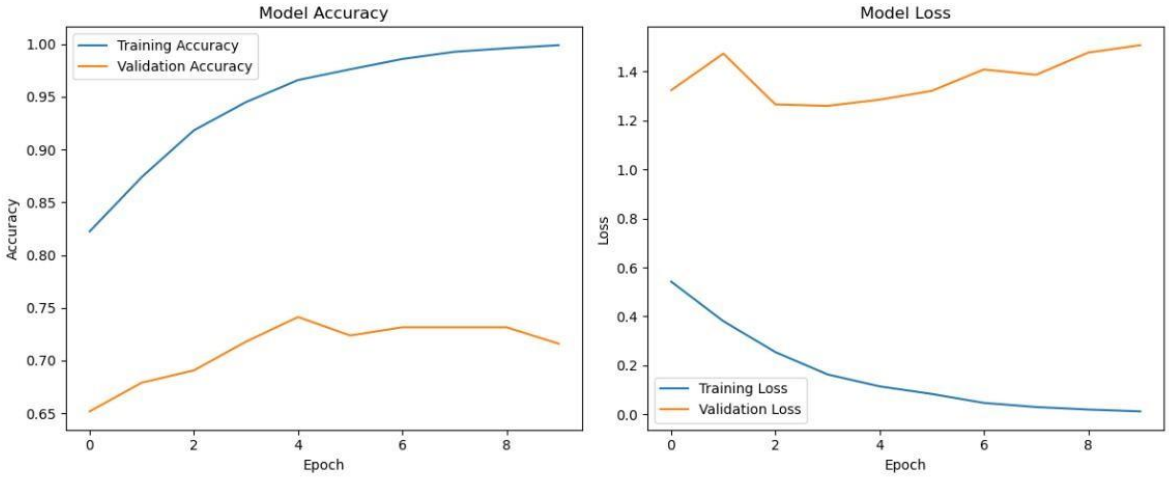
plt.plot(history.history['val_loss'], label='Validation Loss')

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.