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 convolutional neural network with transfer learning.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.
7. Visualize the data, model and accuracy.

Program:

```
import matplotlib.pyplot as plt

import numpy as np

import os

import tensorflow as tf

physical_devices = tf.config.list_physical_devices("GPU")

if len(physical_devices) > 0:

  tf.config.experimental.set_memory_growth(physical_devices[0], True)

# Load the Fashion MNIST dataset

_URL = 'https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip'

zip_path = tf.keras.utils.get_file('cats_and_dogs.zip', origin=_URL, extract=True) # extracting the dataset

PATH = os.path.join(os.path.dirname(zip_path), 'cats_and_dogs_filtered') #path of the files



train_dir = os.path.join(PATH, 'train')

validation_dir = os.path.join(PATH, 'validation')

BATCH_SIZE = 32
```

```
IMG_SIZE = (160, 160)
train_dataset = tf.keras.utils.image_dataset_from_directory(train_dir,
                                                            shuffle=True,
                                                            batch_size=BATCH_SIZE,
                                                            image_size=IMG_SIZE)
validation_dataset = tf.keras.utils.image_dataset_from_directory(validation_dir,
                                                                 shuffle=True,
                                                                 batch_size=BATCH_SIZE,
                                                                 image_size=IMG_SIZE)
val_batches = tf.data.experimental.cardinality(validation_dataset)
test_dataset = validation_dataset.take(val_batches // 5)
validation_dataset = validation_dataset.skip(val_batches // 5)
print('Number of validation batches: %d' % tf.data.experimental.cardinality(validation_dataset))
print('Number of test batches: %d' % tf.data.experimental.cardinality(test_dataset))

AUTOTUNE = tf.data.AUTOTUNE
train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)

data_augmentation = tf.keras.Sequential([
 tf.keras.layers.RandomFlip('horizontal'),
 tf.keras.layers.RandomRotation(0.2),
])
```

```
for image, _ in train_dataset.take(1):
 plt.figure(figsize=(10, 10))
 first_image = image[0]
 for i in range(9):
   ax = plt.subplot(3, 3, i + 1)
   augmented_image = data_augmentation(tf.expand_dims(first_image, 0))
   plt.imshow(augmented_image[0] / 255)
   plt.axis('off')

preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input
rescale = tf.keras.layers.Rescaling(1./127.5, offset=-1)
# Load the pre-trained MobileNetV2 model
IMG_SHAPE = IMG_SIZE + (3,)
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')

image_batch, label_batch = next(iter(train_dataset))
feature_batch = base_model(image_batch)
print(feature_batch.shape)

base_model.trainable = False
base_model.summary()
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
feature_batch_average = global_average_layer(feature_batch)
```

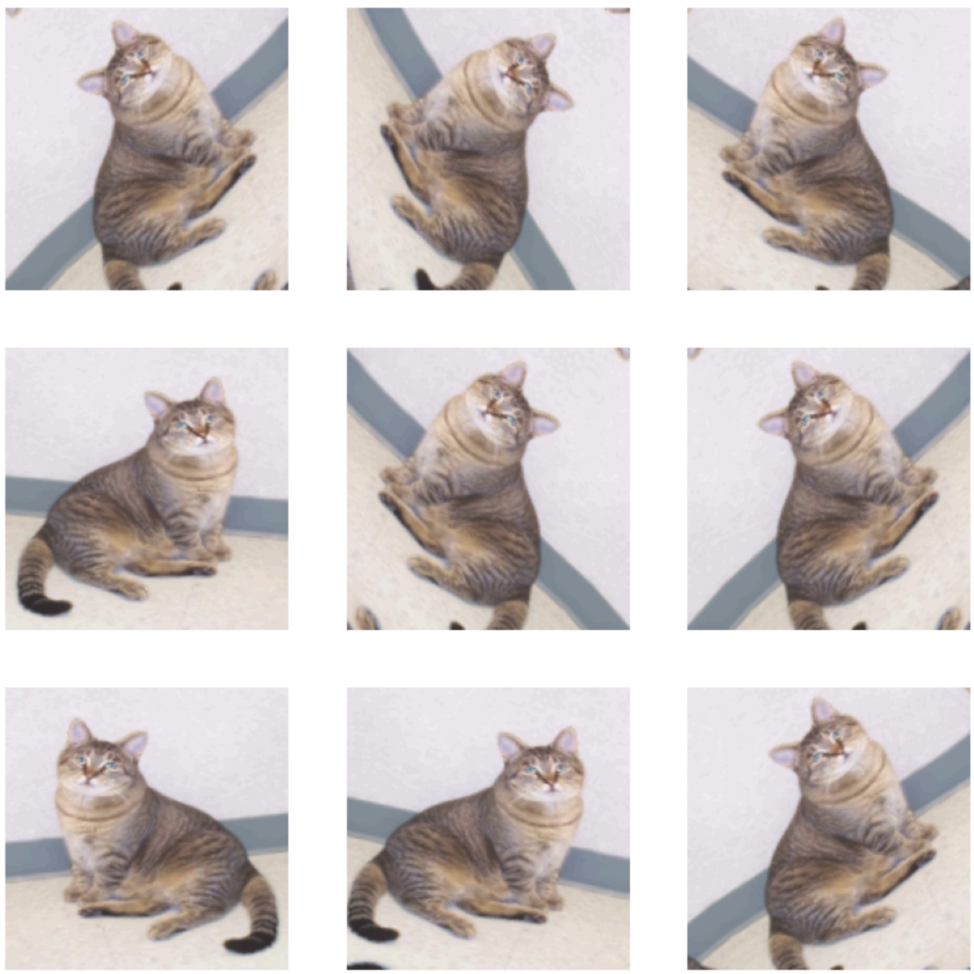
```
print(feature_batch_average.shape)
prediction_layer = tf.keras.layers.Dense(1)
prediction_batch = prediction_layer(feature_batch_average)
print(prediction_batch.shape)
inputs = tf.keras.Input(shape=(160, 160, 3))
x = data_augmentation(inputs)
x = preprocess_input(x)
x = base_model(x, training=False)
x = global_average_layer(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
# Compile the model
base_learning_rate = 0.0001
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
          loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
          metrics=['accuracy'])

# Train the model
history = model.fit(train_dataset,
                epochs=5,
                validation_data=validation_dataset)
```

Output:

```
Downloading data from https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip
68606236/68606236 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Found 2000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

```
Number of validation batches: 26
Number of test batches: 6
```



```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_160_no_top.h5
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
(32, 5, 5, 1280)
```

Model: "mobilenetv2_1.00_160"

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 160, 160, 3)	0	-
Conv1 (Conv2D)	(None, 80, 80, 32)	864	input_layer_1[0][0]
bn_Conv1 (BatchNormalization)	(None, 80, 80, 32)	128	Conv1[0][0]
Conv1_relu (ReLU)	(None, 80, 80, 32)	0	bn_Conv1[0][0]
expanded_conv_depthwise (DepthwiseConv2D)	(None, 80, 80, 32)	288	Conv1_relu[0][0]
expanded_conv_depthwise_… (BatchNormalization)	(None, 80, 80, 32)	128	expanded_conv_depthwi…
expanded_conv_depthwise_… (ReLU)	(None, 80, 80, 32)	0	expanded_conv_depthwi…
expanded_conv_project (Conv2D)	(None, 80, 80, 16)	512	expanded_conv_depthwi…
expanded_conv_project_BN (BatchNormalization)	(None, 80, 80, 16)	64	expanded_conv_project…
block_1_expand (Conv2D)	(None, 80, 80, 96)	1,536	expanded_conv_project…
block_1_expand_BN (BatchNormalization)	(None, 80, 80, 96)	384	block_1_expand[0][0]
block_1_expand_relu (ReLU)	(None, 80, 80, 96)	0	block_1_expand_BN[0][…
block_1_pad (ZeroPadding2D)	(None, 81, 81, 96)	0	block_1_expand_relu[0…
block_1_depthwise	(None, 40, 40, 96)	864	block_1_pad[0][0]

```
Epoch 1/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 65s 919ms/step - accuracy: 0.5219 - loss: 0.8102 - val_accuracy: 0.7030 - val_loss: 0.5759
Epoch 2/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 78s 852ms/step - accuracy: 0.6788 - loss: 0.5871 - val_accuracy: 0.8181 - val_loss: 0.4201
Epoch 3/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 60s 948ms/step - accuracy: 0.7891 - loss: 0.4316 - val_accuracy: 0.8725 - val_loss: 0.3189
Epoch 4/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 78s 881ms/step - accuracy: 0.8389 - loss: 0.3667 - val_accuracy: 0.8936 - val_loss: 0.2581
Epoch 5/5
63/63 ━━━━━━━━━━━━━━━━━━━━ 87s 966ms/step - accuracy: 0.8458 - loss: 0.3303 - val_accuracy: 0.9134 - val_loss: 0.2138
```

Result:

Thus the convolutional neural network with transfer learning and visualization is performed successfully