Exp 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.

Program:

```
#Import TensorFlow

import tensorflow as tf


from tensorflow.keras import datasets, layers, models

import matplotlib.pyplot as plt


#Download and prepare the CIFAR10 dataset

(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()


# Normalize pixel values to be between 0 and 1

train_images, test_images = train_images / 255.0, test_images / 255.0

#Verify the data

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',

        'dog', 'frog', 'horse', 'ship', 'truck']


plt.figure(figsize=(10,10))

for i in range(25):

    plt.subplot(5,5,i+1)

    plt.xticks([])

    plt.yticks([])
```

```python
    plt.grid(False)
    plt.imshow(train_images[i])
    # The CIFAR labels happen to be arrays,
    # which is why you need the extra index
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()


#Create a convulutional base
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.summary()


#Add Dense layers to the top
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
model.summary()


#Evaluate the model
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')


test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

Output:

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_2 (Conv2D)	(None, 4, 4, 64)	36,928
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 64)	65,600
dense_1 (Dense)	(None, 10)	650

```
Total params: 122,570 (478.79 KB)
Trainable params: 122,570 (478.79 KB)
Non-trainable params: 0 (0.00 B)
313/313 - 2s - 6ms/step - accuracy: 0.6860 - loss: 0.9132
```


Output:
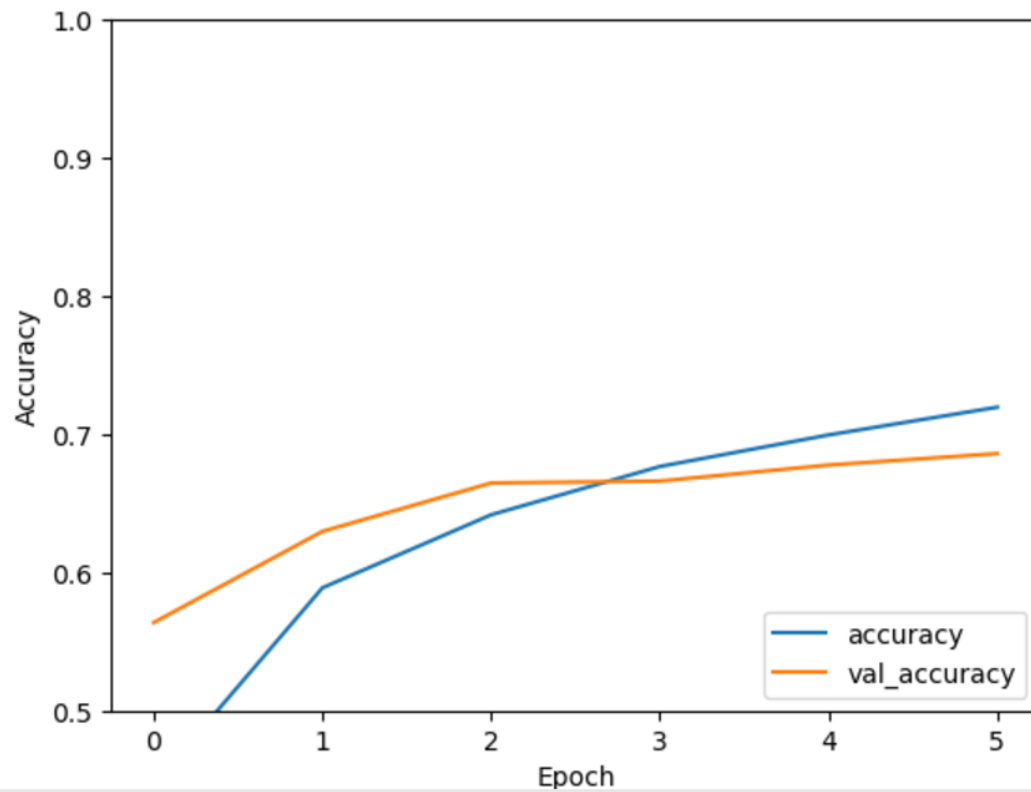
Result:

Therefore, transfer learning and CNN has been Virtualization has been implemented successfully.