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

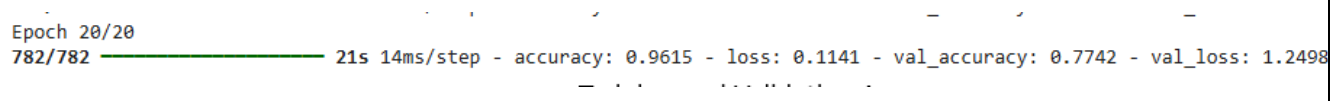
Program:

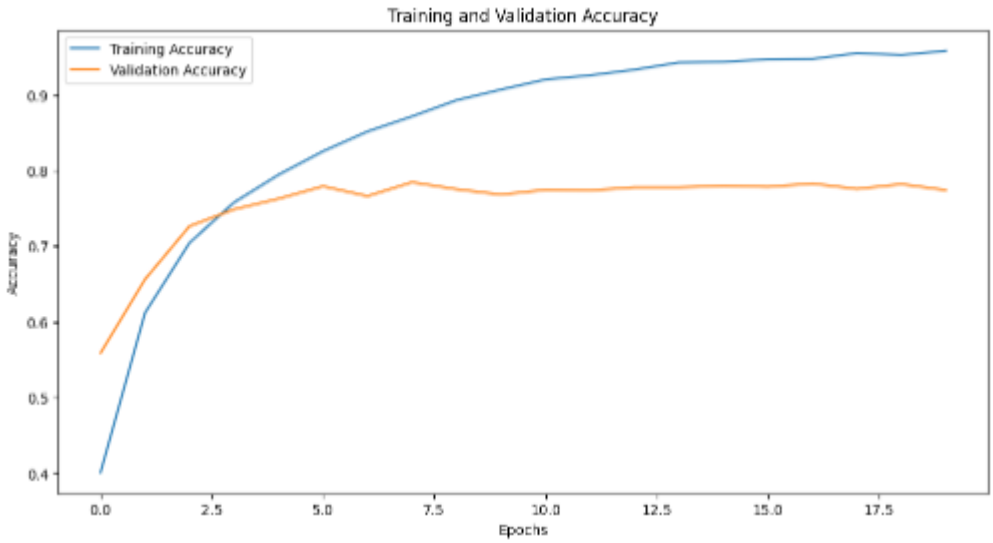
```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import cifar10
import matplotlib.pyplot as plt

# Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalize pixel values to be between 0 and 1
x_train, x_test = x_train / 255.0, x_test / 255.0

# Define the VGG16 model (adjusted for CIFAR-10)
def create_vgg16_model():
    model = models.Sequential([
        # VGG16 architecture adapted for smaller input size (32x32)
```

```
        layers.InputLayer(input_shape=(32, 32, 3)),

        # Convolutional Block 1
        layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),

        # Convolutional Block 2
        layers.Conv2D(128, (3, 3), padding='same', activation='relu'),
        layers.Conv2D(128, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),

        # Convolutional Block 3
        layers.Conv2D(256, (3, 3), padding='same', activation='relu'),
        layers.Conv2D(256, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),

        # Fully connected layers
        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(10, activation='softmax')  # 10 classes for CIFAR-10
    ])

    return model

# Create the model
```

```
model = create_vgg16_model()


# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])


# Train the model
history = model.fit(x_train, y_train, epochs=20, validation_data=(x_test, y_test),
batch_size=64)


# Plot the accuracy
plt.figure(figsize=(12, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()


# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"Test Accuracy: {test_acc:.4f}")
```

Output:





Result:

Thus to build a convolutional neural network with transfer learning and perform visualization was completed successfully.