PYTHON PROGRAM TO VISUALIZE AND DESIGN CNN WITH TRANSFER LEARNING

Aim:

To visualize and design CNN with transfer learning in python.

Procedure:

1. Import TensorFlow, CIFAR-10 dataset, VGG16 model, and necessary Keras layers for building the model.

2. Load the CIFAR-10 dataset, consisting of 60,000 images, and divide it into training and test sets.

3. Normalize the training and test images by scaling pixel values to the range [0, 1].

4. One-hot encode the training and test labels to convert them into categorical format.

5. Load the pre-trained VGG16 model with ImageNet weights, excluding the top fully connected layers.

6. Freeze the layers of the pre-trained VGG16 model to prevent them from being trained.

7. Initialize a Sequential model and add the pre-trained VGG16 as the base.

8. Add a Flatten layer followed by a Dense layer with 256 units and ReLU activation.

9. Add a final Dense layer with 10 units and softmax activation for CIFAR-10 classification.

10. Compile the model with Adam optimizer, train for 10 epochs, and evaluate it on test data to print the accuracy.

Code:

```
import tensorflow as tf
```

```python
from tensorflow.keras.datasets import cifar10 from
tensorflow.keras.applications import VGG16 from
tensorflow.keras.models import Sequential from
tensorflow.keras.layers import Flatten, Dense from
tensorflow.keras.utils import to_categorical

# Load CIFAR-10 dataset
(itrain, ltrain), (itest, ltest) = cifar10.load_data()

# Preprocess the data itrain =
itrain / 255.0 itest = itest /
255.0          ltrain          =
to_categorical(ltrain) ltest =
to_categorical(ltest)

# Load pre-trained VGG16 model (excluding the top fully-connected layers) basem =
VGG16(weights='imagenet', include_top=False, input_shape=(32, 32, 3))

# Freeze the pre-trained layers for
layer in basem.layers:
    layer.trainable = False

# Create a new model on top semodel
= Sequential() semodel.add(basem)
semodel.add(Flatten())
semodel.add(Dense(256, activation='relu'))
semodel.add(Dense(10, activation='softmax')) # CIFAR-10 has 10 classes
# Compile the model
semodel.compile(optimizer='adam',                          loss='categorical_crossentropy',
metrics=['accuracy'])
```

# Train the model
semodel.fit(itrain, ltrain, epochs=10, batch_size=32, validation_data=(itest, ltest))

Evaluate the model on test data ltest, atest
= semodel.evaluate(itest, ltest) print("Test
accuracy:", atest)

Output:

```
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━ 121s 77ms/step - accuracy: 0.4755 - loss: 1.4946 - val_accuracy: 0.5627 - val_loss: 1.2390
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━ 127s 81ms/step - accuracy: 0.5919 - loss: 1.1764 - val_accuracy: 0.5817 - val_loss: 1.1992
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━ 129s 82ms/step - accuracy: 0.6114 - loss: 1.1064 - val_accuracy: 0.5908 - val_loss: 1.1614
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━ 125s 80ms/step - accuracy: 0.6290 - loss: 1.0572 - val_accuracy: 0.6099 - val_loss: 1.1226
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━ 123s 79ms/step - accuracy: 0.6389 - loss: 1.0196 - val_accuracy: 0.6101 - val_loss: 1.1246
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━ 122s 78ms/step - accuracy: 0.6570 - loss: 0.9742 - val_accuracy: 0.6099 - val_loss: 1.1118
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━ 121s 77ms/step - accuracy: 0.6702 - loss: 0.9425 - val_accuracy: 0.6192 - val_loss: 1.1128
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━ 115s 74ms/step - accuracy: 0.6810 - loss: 0.9023 - val_accuracy: 0.6108 - val_loss: 1.1179
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━ 116s 74ms/step - accuracy: 0.6898 - loss: 0.8815 - val_accuracy: 0.6194 - val_loss: 1.1053
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━ 115s 73ms/step - accuracy: 0.6964 - loss: 0.8575 - val_accuracy: 0.6141 - val_loss: 1.1101
313/313 ━━━━━━━━━━━━━━━ 19s 61ms/step - accuracy: 0.6153 - loss: 1.1069
Test accuracy: 0.6140999794006348
```

Result:

     Thus, to visualize and design CNN with transfer learning has been completed successfully.