In [ ]:

```
import numpy as np
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from tensorflow.keras.models import load_model

# 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_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# One-hot encode labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# Split the data into training and testing sets
X_train, X_val, y_train, y_val = train_test_split(
    X_train, y_train, test_size=0.1, random_state=42)

# Load pre-trained VGG16 model
base_model = VGG16(weights='imagenet', include_top=False,
                   input_shape=(32, 32, 3))

# Freeze convolutional layers
for layer in base_model.layers:
    layer.trainable = False

# Create a new model by adding custom dense layers on top of VGG16
x = Flatten()(base_model.output)
x = Dense(64, activation='relu')(x)
output = Dense(10, activation='softmax')(x)

# load the saved model
try:
    model = load_model('model.h5')
except:
    model = None
# if not found then create a new model
if model is None:
    model = Model(inputs=base_model.input, outputs=output)

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

# Train the model
model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val))
```

```python
# Evaluate the model
y_pred = np.argmax(model.predict(X_test), axis=1)
y_true = np.argmax(y_test, axis=1)

# save the model
model.save('model.h5')

accuracy = accuracy_score(y_true, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Epoch 1/10
1407/1407 [==============================] - 175s 124ms/step - loss: 1.0237 - accura
cy: 0.6422 - val_loss: 1.1419 - val_accuracy: 0.6000
Epoch 2/10
1407/1407 [==============================] - 182s 130ms/step - loss: 1.0119 - accura
cy: 0.6436 - val_loss: 1.1327 - val_accuracy: 0.6042
Epoch 3/10
1407/1407 [==============================] - 181s 128ms/step - loss: 0.9990 - accura
cy: 0.6516 - val_loss: 1.1439 - val_accuracy: 0.6004
Epoch 4/10
1407/1407 [==============================] - 179s 127ms/step - loss: 0.9897 - accura
cy: 0.6550 - val_loss: 1.1284 - val_accuracy: 0.6088
Epoch 5/10
1407/1407 [==============================] - 179s 127ms/step - loss: 0.9784 - accura
cy: 0.6586 - val_loss: 1.1291 - val_accuracy: 0.6068
Epoch 6/10
1407/1407 [==============================] - 180s 128ms/step - loss: 0.9704 - accura
cy: 0.6599 - val_loss: 1.1413 - val_accuracy: 0.6038
Epoch 7/10
1407/1407 [==============================] - 180s 128ms/step - loss: 0.9598 - accura
cy: 0.6653 - val_loss: 1.1204 - val_accuracy: 0.6126
Epoch 8/10
1407/1407 [==============================] - 185s 132ms/step - loss: 0.9494 - accura
cy: 0.6673 - val_loss: 1.1400 - val_accuracy: 0.6060
Epoch 9/10
1407/1407 [==============================] - 184s 131ms/step - loss: 0.9421 - accura
cy: 0.6708 - val_loss: 1.1304 - val_accuracy: 0.6058
Epoch 10/10
1407/1407 [==============================] - 192s 137ms/step - loss: 0.9333 - accura
cy: 0.6743 - val_loss: 1.1333 - val_accuracy: 0.6084
313/313 [==============================] - 38s 121ms/step
Accuracy: 60.05%
```

```
c:\Users\rohit\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\e
ngine\training.py:3000: UserWarning: You are saving your model as an HDF5 file via `
model.save()`. This file format is considered legacy. We recommend using instead the
native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```