Load Google Drive

```
# Add google drive
!apt-get install -y -qq software-properties-common python-software-properties module-i
!add-apt-repository -y ppa:alessandro-strada/ppa 2>&1 > /dev/null
!apt-get update -qq 2>&1 > /dev/null
!apt-get -y install -qq google-drive-ocamlfuse fuse
from google.colab import auth
auth.authenticate_user()
from oauth2client.client import GoogleCredentials
creds = GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret}
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.
!mkdir -p drive
!google-drive-ocamlfuse drive
```

```
    E: Package 'python-software-properties' has no installation candidate
    Selecting previously unselected package google-drive-ocamlfuse.
    (Reading database ... 145483 files and directories currently installed.)
    Preparing to unpack .../google-drive-ocamlfuse_0.7.23-0ubuntu1~ubuntu18.04.1_amd
    Unpacking google-drive-ocamlfuse (0.7.23-0ubuntu1~ubuntu18.04.1) ...
    Setting up google-drive-ocamlfuse (0.7.23-0ubuntu1~ubuntu18.04.1) ...
    Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
    Please, open the following URL in a web browser: https://accounts.google.com/o/oa
    ..........
    Please, open the following URL in a web browser: https://accounts.google.com/o/oa
    Please enter the verification code: Access token retrieved correctly.
```

Navigate

to the folder containing data and makedata python file

Location of python file and data is important, but can be modified!

```
!pwd
```

```
    /content
```

```
!ls
```

```
    adc.json  drive  sample_data
```

```
cd drive/DataSets/CIFAR
```

```
    /content/drive/DataSets/CIFAR
```

```
!pwd
```

```
    /content/drive/DataSets/CIFAR
```

Imports

```
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Dense, Flatten, Conv2D
from tensorflow.keras import Model
import numpy as np

import makedata        # to get the CIFAR10 data in the required format
```

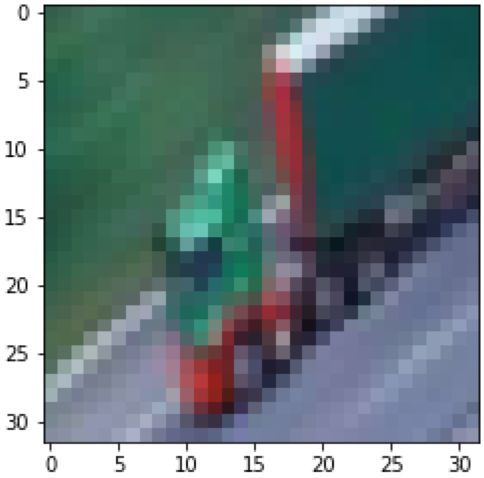
```
# Load data
x_train, y_train, x_test, y_test, a, b = makedata.cifar10()
```

```
    loaded data_batch_1
    len of data_batch_1 :  10000
    len of training data  10000
    ================
    loaded data_batch_2
    len of data_batch_2 :  10000
    len of training data  20000
    ================
    loaded data_batch_3
    len of data_batch_3 :  10000
    len of training data  30000
    ================
    loaded data_batch_4
    len of data_batch_4 :  10000
    len of training data  40000
    ================
    loaded data_batch_5
    len of data_batch_5 :  10000
    len of training data  50000
    ================
    ==============================
    full data info:
    x_train shape: (50000, 32, 32, 3)
    y_train shape: (50000, 10)
    x_test shape:  (10000, 32, 32, 3)
    y_test shape: (10000, 10)
```

```
# check data
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import matplotlib.image as mpimg

im = x_train[50]
%pylab inline
imgplot = plt.imshow(im)
plt.show()
```

```
     Populating the interactive namespace from numpy and matplotlib
```



```
# Normalize data
x_train = x_train/255
x_test = x_test/255
```

```
input_shape=(32,32,3)
img_input = tf.keras.layers.Input(shape=input_shape)
def VGGmodel():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))
    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x




model = VGGmodel()
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 4, 4, 512)         1180160
```

```
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 4, 4, 512)         2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 4, 4, 512)         2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 2, 2, 512)         0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 2, 2, 512)         2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 2, 2, 512)         2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 2, 2, 512)         2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 512)               0         
_________________________________________________________________
fc1 (Dense)                  (None, 40)                20520     
_________________________________________________________________
fc2 (Dense)                  (None, 40)                1640      
_________________________________________________________________
predictions (Dense)          (None, 10)                410       
=================================================================
Total params: 14,737,258
Trainable params: 14,737,258
Non-trainable params: 0
_________________________________________________________________
```

```
#Validate


model = VGGmodel()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)



# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 13s 51ms/step - loss: 1.0192e-04 - accu
test loss, test acc: [0.00010164660488953814, 1.0]
Evaluate on test data
20/20 [==============================] - 1s 74ms/step - loss: 3.6595 - accuracy:
test loss, test acc: [3.6595301628112793, 0.6635000109672546]
```

Test Momoriation Layer Wise

reinitialization layer by layer (one at a time)

```
# reinitialize block1_conv1
def one():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))
    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x
```

```
model = one()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)


# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 5s 45ms/step - loss: 10.0084 - accuracy
test loss, test acc: [10.035187721252441, 0.10261999815702438]
Evaluate on test data
20/20 [==============================] - 1s 43ms/step - loss: 9.9862 - accuracy:
test loss, test acc: [9.986235618591309, 0.10379999876022339]
```

```
# Evaluate the model on the train data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 4s 44ms/step - loss: 10.0352 - accuracy
test loss, test acc: [10.035187721252441, 0.10261999815702438]
```

```
# How does dropout and normalization affect the contribution of individual layers in r
```

reinitialize 2nd Conv layer

```
# reinitialize block1_conv2
def two():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))
    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x




model = two()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)


# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
```

```
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 5s 44ms/step - loss: 6.0817 - accuracy:
test loss, test acc: [6.087360858917236, 0.11794000118970871]
Evaluate on test data
20/20 [==============================] - 1s 43ms/step - loss: 6.1021 - accuracy:
test loss, test acc: [6.102051258087158, 0.11909999698400497]
```

reinitialize 3rd Conv Layer

```
# reinitialize block2_conv1
def three():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))
    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x




model = three()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)


# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 5s 45ms/step - loss: 4.7583 - accuracy:
test loss, test acc: [4.7585530281066895, 0.09504000097513199]
Evaluate on test data
20/20 [==============================] - 1s 44ms/step - loss: 4.7916 - accuracy:
test loss, test acc: [4.791600227355957, 0.09489999711513519]
```

reinitialize 4th layer

```
# reinitialize block2_conv2
def four():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))
    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x




model = four()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)



# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 5s 45ms/step - loss: 3.3857 - accuracy:
test loss, test acc: [3.3937768936157227, 0.12678000330924988]
```

```
Evaluate on test data
20/20 [==============================] - 1s 43ms/step - loss: 3.3837 - accuracy:
test loss, test acc: [3.3836557865142822, 0.12880000472068787]
```

reinitialize 5th layer

```
# reinitialize block3_conv1
def five():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))
    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


    #compile the model
    x.compile(loss='categorical_crossentropy', optimizer='SGD', metrics=['accuracy'])
    return x
```

```
model = five()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)


# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 5s 45ms/step - loss: 4.3921 - accuracy:
test loss, test acc: [4.40025520324707, 0.10608000308275223]
Evaluate on test data
20/20 [==============================] - 1s 44ms/step - loss: 4.3940 - accuracy:
test loss, test acc: [4.394033908843994, 0.10610000044107437]
```

reinitialize 6th layer

```
# reinitialize block3_conv2
def six():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))
    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))

    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x
```

```
model = six()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)


# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 5s 45ms/step - loss: 4.6020 - accuracy:
test loss, test acc: [4.612954616546631, 0.10322000086307526]
Evaluate on test data
20/20 [==============================] - 1s 44ms/step - loss: 4.6570 - accuracy:
test loss, test acc: [4.6570048332214355, 0.10220000147819519]
```

reinitialize 7th Layer

```
# reinitialize block3_conv3
def seven():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))

    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x



model = seven()
model = seven()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)


# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 5s 45ms/step - loss: 4.6456 - accuracy:
test loss, test acc: [4.643528461456299, 0.1151999980211258]
Evaluate on test data
20/20 [==============================] - 1s 45ms/step - loss: 4.6431 - accuracy:
test loss, test acc: [4.643050193786621, 0.11710000038146973]
```

reinitialize 8th layer

```
# reinitialize block4_conv1
def eight():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))
    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x




model = eight()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)


# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 5s 45ms/step - loss: 5.7014 - accuracy:
test loss, test acc: [5.71537446975708, 0.14127999544143677]
Evaluate on test data
20/20 [==============================] - 1s 45ms/step - loss: 5.7030 - accuracy:
test loss, test acc: [5.702980995178223, 0.14229999482631683]
```

reinitialize 9th layer

```
# reinitialize block4_conv2
def nine():
    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))

    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x




model = nine()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)


# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 5s 45ms/step - loss: 8.9731 - accuracy:
test loss, test acc: [8.998225212097168, 0.08488000184297562]
Evaluate on test data
20/20 [==============================] - 1s 45ms/step - loss: 8.9997 - accuracy:
test loss, test acc: [8.999748229980469, 0.08449999988079071]
```

reinitialize 10th layer

```
# reinitialize block4_conv3
def ten():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))
    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x




model = ten()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)


# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 5s 45ms/step - loss: 22.3507 - accuracy
test loss, test acc: [22.400789260864258, 0.11286000162363052]
Evaluate on test data
20/20 [==============================] - 1s 44ms/step - loss: 22.4948 - accuracy
test loss, test acc: [22.494808197021484, 0.11079999804496765]
```

reinitialize 11th layer

```
# reinitialize block5_conv1
def eleven():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))

    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x




model = eleven()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)



# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
```

```
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
    Evaluate on train data
    98/98 [==============================] - 5s 46ms/step - loss: 9.7984 - accuracy:
    test loss, test acc: [9.82347297668457, 0.10409999638795853]
    Evaluate on test data
    20/20 [==============================] - 1s 45ms/step - loss: 9.8875 - accuracy:
    test loss, test acc: [9.887468338012695, 0.10369999706745148]
```

reinitialize 12th layer

```
# reinitialize block5_conv2
def twelve():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))

    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x




model = twelve()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)


# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
98/98 [==============================] - 5s 46ms/step - loss: 5.7248 - accuracy:
test loss, test acc: [5.7239813804626465, 0.10118000209331512]
Evaluate on test data
20/20 [==============================] - 1s 44ms/step - loss: 5.7073 - accuracy:
test loss, test acc: [5.707275867462158, 0.10119999945163727]
```

reinitialize 13th layer

```
# reinitialize block5_conv3
def thirteen():

    x = tf.keras.Sequential()

    # Block 1
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', name='
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool'))

    # Block 2
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool'))

    # Block 3
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool'))

    # Block 4
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool'))

    # Block 5
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.Conv2D(512, (3, 3), activation='relu', padding='same', name=
    x.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool'))


    # Classification block
    x.add(tf.keras.layers.Flatten(name='flatten'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc1'))
    x.add(tf.keras.layers.Dense(40, activation='relu', name='fc2'))
    x.add(tf.keras.layers.Dense(10, activation='softmax', name='predictions'))


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

    return x




model = thirteen()
model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)

# Evaluate the model on the test data using `evaluate`
print("Evaluate on train data")
results = model.evaluate(x_train, y_train, batch_size=512)
print("test loss, test acc:", results)


# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=512)
print("test loss, test acc:", results)
```

```
Evaluate on train data
```

```
98/98 [==============================] - 5s 46ms/step - loss: 5.0734 - accuracy:
test loss, test acc: [5.060662746429443, 0.0987199991941452]
Evaluate on test data
20/20 [==============================] - 1s 45ms/step - loss: 5.0783 - accuracy:
test loss, test acc: [5.078311920166016, 0.09700000286102295]
```