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 ... 145480 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/o
    ..........
    Please, open the following URL in a web browser: https://accounts.google.com/o/o
    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
```

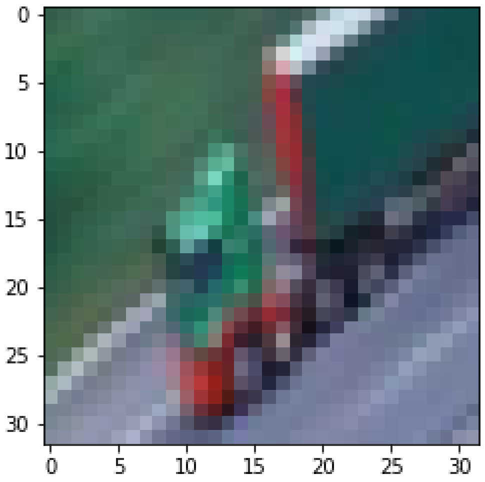
```
# Load data
x_train, y_train, x_test, y_test, a, b = makedata.cifar10()
```

```
    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
_________________________________________________________________
```

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

model.fit(x_train,
         y_train,
         batch_size=512,
         epochs=250,
         validation_data=(x_test, y_test))
```

```
Epoch 1/250
98/98 [==============================] - 28s 185ms/step - loss: 2.3026 - accurac
Epoch 2/250
98/98 [==============================] - 16s 165ms/step - loss: 2.3025 - accurac
Epoch 3/250
98/98 [==============================] - 16s 159ms/step - loss: 2.3025 - accurac
Epoch 4/250
98/98 [==============================] - 16s 161ms/step - loss: 2.3024 - accurac
Epoch 5/250
98/98 [==============================] - 16s 164ms/step - loss: 2.3024 - accurac
Epoch 6/250
98/98 [==============================] - 17s 170ms/step - loss: 2.3023 - accurac
Epoch 7/250
98/98 [==============================] - 16s 167ms/step - loss: 2.3023 - accurac
Epoch 8/250
98/98 [==============================] - 17s 170ms/step - loss: 2.3022 - accurac
Epoch 9/250
98/98 [==============================] - 17s 172ms/step - loss: 2.3021 - accurac
Epoch 10/250
```

```
    98/98 [==============================] - 18s 180ms/step - loss: 2.3020 - accurac
    Epoch 11/250
    98/98 [==============================] - 17s 176ms/step - loss: 2.3020 - accurac
    Epoch 12/250
    98/98 [==============================] - 17s 179ms/step - loss: 2.3018 - accurac
    Epoch 13/250
    98/98 [==============================] - 17s 176ms/step - loss: 2.3018 - accurac
    Epoch 14/250
    98/98 [==============================] - 18s 181ms/step - loss: 2.3016 - accurac
    Epoch 15/250
    98/98 [==============================] - 17s 175ms/step - loss: 2.3015 - accurac
    Epoch 16/250
    98/98 [==============================] - 17s 176ms/step - loss: 2.3013 - accurac
    Epoch 17/250
    98/98 [==============================] - 17s 176ms/step - loss: 2.3011 - accurac
    Epoch 18/250
    98/98 [==============================] - 18s 182ms/step - loss: 2.3009 - accurac
    Epoch 19/250
    98/98 [==============================] - 17s 176ms/step - loss: 2.3006 - accurac
    Epoch 20/250
    98/98 [==============================] - 17s 175ms/step - loss: 2.3001 - accurac
    Epoch 21/250
    98/98 [==============================] - 17s 177ms/step - loss: 2.2997 - accurac
    Epoch 22/250
    98/98 [==============================] - 18s 182ms/step - loss: 2.2990 - accurac
    Epoch 23/250
    98/98 [==============================] - 17s 175ms/step - loss: 2.2982 - accurac
    Epoch 24/250
    98/98 [==============================] - 17s 175ms/step - loss: 2.2971 - accurac
    Epoch 25/250
    98/98 [==============================] - 17s 176ms/step - loss: 2.2956 - accurac
    Epoch 26/250
    98/98 [==============================] - 18s 181ms/step - loss: 2.2933 - accurac
    Epoch 27/250
    98/98 [==============================] - 17s 176ms/step - loss: 2.2901 - accurac
    Epoch 28/250
    98/98 [==============================] - 17s 176ms/step - loss: 2.2850 - accurac
    Epoch 29/250
    98/98 [==============================] - 17s 176ms/step - loss: 2.2768 - accurac
    Epoch 30/250
```

```
model.save_weights("Weights/MemorizeCIFAR10VGG16")
model.save_weights("Weights/MemorizeCIFAR10VGG16.h5")
```

```
#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 [==============================] - 5s 48ms/step - loss: 1.0192e-04 - accur
test loss, test acc: [0.00010164660488953814, 1.0]
Evaluate on test data
20/20 [==============================] - 1s 47ms/step - loss: 3.6595 - accuracy:
test loss, test acc: [3.6595301628112793, 0.6635000109672546]
```

Test Momoriation Layer Wise

```
# removing block1_conv1
# adding input to block1_conv2
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)
```

```
      1 model = one()
----> 2 model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)
      3
      4 # Evaluate the model on the test data using `evaluate`
      5 print("Evaluate on train data")
```

1 frames

```
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/keras/saving/hdf5_format.py in
load_weights_from_hdf5_group_by_name(f, layers, skip_mismatch)
    786                                 symbolic_weights[i])) +
    787                             ', but the saved weight has shape ' +
--> 788                             str(weight_values[i].shape) + '.')
    789
    790         else:

ValueError: Layer #1 (named "block1_conv2"), weight <tf.Variable
'block1_conv2/kernel:0' shape=(3, 3, 3, 64) dtype=float32, numpy=
array([[[[-0.09956303, -0.0581462 ,  0.0248725 , ..., -0.07316834,
          -0.0387677 , -0.0537156 ],
         [-0.07761978, -0.09973869, -0.09407175, ..., -0.00862814,
          -0.064025  ,  0.08160579],
         [-0.01617178,  0.08721732,  0.0827115 , ..., -0.09563942,
           0.04142563, -0.04711454]],

        [[ 0.06843044, -0.04180246,  0.05545586, ..., -0.01387551,
           0.06913456,  0.01329578],
         [ 0.01045016,  0.08218001, -0.00681939, ...,  0.05114198,
          -0.04527047,  0.06545711],
         [-0.02201322, -0.01965424, -0.03502738, ...,  0.07616936,
           0.01475172, -0.0261196 ]],

        [[ 0.01650464,  0.06005917, -0.0458339 , ..., -0.06912289,
          -0.09243204,  0.0706805 ],
         [-0.00254123, -0.08432993, -0.09504917, ...,  0.09500003,
          -0.06290736, -0.03277694],
         [ 0.01637642, -0.01344828,  0.01203163, ...,  0.0410036 ,
          -0.03086925, -0.08723149]]],


       [[[ 0.06865628,  0.05378024, -0.05542901, ..., -0.06530946,
          -0.03176162, -0.07896222],
         [-0.02446786, -0.02049828,  0.06662394, ...,  0.07758683,
          -0.07503402, -0.00456043],
         [-0.06432889,  0.00971234, -0.06161709, ...,  0.06961535,
          -0.08641529, -0.0279329 ]],

        [[-0.06768909, -0.00811768, -0.09333155, ..., -0.07953781,
           0.03972922,  0.02792588],
         [-0.03557428, -0.0376708 ,  0.08457108, ..., -0.04090034,
          -0.06538101, -0.01338948],
         [ 0.09875098, -0.05389019,  0.04817709, ...,  0.09668204,
           0.07487325, -0.02674799]],

        [[-0.07492603,  0.05432467, -0.04895275, ..., -0.00026834,
          -0.00097851, -0.03360571],
         [-0.09180516, -0.09633549, -0.08066   , ..., -0.08444801,
          -0.09646648,  0.06713296],
              [-0.05136765,  0.03555307, -0.00559523, ..., -0.07593821,
                0.08852868,  0.00826082]]],


# 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)
               -0.03660446,  0.04444617]],

# How does dropout and normalization affect the contribution of individual layers in m
                0.00174155,  0.00435404],


              [-0.07722279, -0.05575871,  0.04837921, ..., -0.0778821 ,
```

Skip 2nd Conv layer

```
              [-0.04384292, -0.04976854,  0.02312082, ...,  0.05667195,

# removing 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 [==============================] - 4s 41ms/step - loss: 6.5875 - accuracy:
test loss, test acc: [6.6006999015808105, 0.13242000341415405]
Evaluate on test data
20/20 [==============================] - 1s 40ms/step - loss: 6.6344 - accuracy:
test loss, test acc: [6.634420871734619, 0.1331000030040741]
```

Skpi 3rd Conv Layer

```
# removing 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)
```

```
      1 model = three()
----> 2 model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)
      3 
      4 # Evaluate the model on the test data using `evaluate`
      5 print("Evaluate on train data")
```

1 frames

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/saving/hdf5_format.py in
load_weights_from_hdf5_group_by_name(f, layers, skip_mismatch)
    786                           symbolic_weights[i])) +
    787                       ', but the saved weight has shape ' +
--> 788                       str(weight_values[i].shape) + '.')
    789 
    790         else:

ValueError: Layer #4 (named "block2_conv2"), weight <tf.Variable
'block2_conv2/kernel:0' shape=(3, 3, 64, 128) dtype=float32, numpy=
array([[[[-0.04802071,  0.01295354,  0.0030825 , ...,  0.02429712,
           0.05703909,  0.02874745],
         [-0.03954303, -0.03420591, -0.01982086, ...,  0.05403095,
           0.00045313, -0.03244872],
         [-0.0162567 , -0.05163736, -0.0208768 , ...,  0.01132591,
           0.04177297, -0.03363217],
         ...,
         [-0.04135139, -0.01727444,  0.00169349, ...,  0.02310072,
           0.00210651, -0.00715797],
         [-0.05860644,  0.00279303,  0.0238493 , ..., -0.02094472,
          -0.01647276,  0.04921841],
         [ 0.0574255 ,  0.00928648, -0.00125413, ...,  0.00665351,
           0.0338912 ,  0.0285783 ]],

        [[-0.00190644, -0.05263561, -0.04026991, ...,  0.00493704,
           0.01236688, -0.03061074],
         [ 0.02642349, -0.01526945, -0.00232701, ...,  0.0096574 ,
          -0.01576785, -0.04596122],
         [ 0.00785022,  0.00018473, -0.00050719, ..., -0.01701339,
          -0.01072202, -0.04465963],
         ...,
         [ 0.03532805, -0.0261611 ,  0.01289174, ..., -0.00443929,
           0.03315306, -0.03825691],
         [-0.04291578, -0.03218305, -0.01793833, ..., -0.01744951,
          -0.02142246, -0.01942721],
         [-0.05631451,  0.05270101, -0.05482937, ..., -0.04287446,
          -0.00677794, -0.01894518]],

        [[ 0.05829093,  0.05102137, -0.03488088, ...,  0.01158215,
          -0.05330763, -0.00646198],
         [-0.05749432,  0.02636534, -0.02706745, ...,  0.05872351,
           0.01957998,  0.04711365],
         [-0.00898276, -0.04207235,  0.05632596, ..., -0.02619477,
          -0.0042291 ,  0.00358532],
         ...,
         [ 0.03417809,  0.01438009, -0.04970001, ..., -0.0119817 ,
           0.01666038, -0.01803104],
         [ 0.03794471, -0.03324047,  0.05656892, ..., -0.03071093,
           0.04814741, -0.01784288],
         [ 0.0470206 , -0.02698814,  0.03307076, ..., -0.01269282,
          -0.01904273,  0.01184687]]],


       [[[ 0.03128475,  0.03637521, -0.0107952 , ..., -0.05765206,
           0.01102645, -0.01894689],
         [-0.0185947 ,  0.01799083,  0.0456076 , ...,  0.05738884,
          -0.02600886,  0.04630667],
         [-0.01994362, -0.04257838,  0.00500234, ...,  0.04595035,
          -0.00032297,  0.00225511],
         ...,
         [-0.00421464,  0.03868308,  0.02718587, ...,  0.02514537,
          -0.05321786,  0.04645413],
         [ 0.01657062,  0.05460292,  0.04033677, ...,  0.03804038,
          -0.034428  , -0.04253592],
         [ 0.03426475, -0.0394787 , -0.01467758, ..., -0.03400338,
          -0.01014201, -0.02203902]],

        [[-0.00833273,  0.03060601, -0.01778496, ...,  0.04342743,
          -0.05513661, -0.03655076],
         [-0.04249819, -0.02334074,  0.03872233, ..., -0.00638324,
          -0.01944479,  0.03972403],
         [ 0.04183294,  0.03063439,  0.03749109, ...,  0.03499331,
          -0.04342103,  0.01600146],
         ...,
         [-0.02753593, -0.01233895,  0.01575123, ...,  0.01190614,
          -0.01548887, -0.02577098],
         [-0.0338449 ,  0.05702272,  0.02837231, ..., -0.02506153,
           0.0224803 ,  0.01061384],
         [-0.01958262, -0.04695094,  0.00885757, ..., -0.04261275,
          -0.04001505,  0.0071568 ]],

        [[-0.02724139,  0.02221139,  0.02811217, ...,  0.03306517,
          -0.03296924,  0.05222968],
         [ 0.00027082, -0.02969885,  0.05885386, ...,  0.05360835,
          -0.02651943, -0.00433684],
         [-0.04725968,  0.03123834,  0.03459484, ..., -0.00253813,
           0.0265805 ,  0.04079583],
         ...,
```

skpi 4th layer

```
          [ 0.00978453, -0.05711376, -0.03697587, ..., -0.01174185,
```

```
# removing 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'))

    # 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 43ms/step - loss: 3.8511 - accuracy:
test loss, test acc: [3.8512916564941406, 0.16176000237464905]
Evaluate on test data
20/20 [==============================] - 1s 42ms/step - loss: 3.8529 - accuracy:
test loss, test acc: [3.8528647422790527, 0.16459999978542328]
```

Skip 5th layer

```
# removing 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)
```

```
      1 model = five()
----> 2 model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)
      3 
      4 # Evaluate the model on the test data using `evaluate`
      5 print("Evaluate on train data")
```

1 frames

```
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/keras/saving/hdf5_format.py in
load_weights_from_hdf5_group_by_name(f, layers, skip_mismatch)
    786                                  symbolic_weights[i])) +
    787                             ', but the saved weight has shape ' +
--> 788                             str(weight_values[i].shape) + '.')
    789 
    790         else:

ValueError: Layer #7 (named "block3_conv2"), weight <tf.Variable
'block3_conv2/kernel:0' shape=(3, 3, 128, 256) dtype=float32, numpy=
array([[[[-0.04001279, -0.00713868, -0.01921073, ..., -0.01519316,
          -0.02375777,  0.02264925],
         [ 0.02013601, -0.01898415, -0.03004941, ...,  0.0004374 ,
          -0.04052394,  0.01721628],
         [ 0.03613825, -0.01155077,  0.01095703, ...,  0.03641537,
           0.0253948 , -0.00955537],
         ...,
         [-0.02730735, -0.00736086, -0.01391627, ...,  0.01131168,
           0.0231651 , -0.02994284],
         [ 0.02562502,  0.01056615, -0.03487313, ..., -0.02168141,
           0.00985314,  0.00457846],
         [-0.02639409,  0.03751725, -0.03158437, ...,  0.0099826 ,
          -0.02714246,  0.04068251]],

        [[-0.01603894, -0.02037246,  0.04128297, ...,  0.03736879,
           0.02496584, -0.0312712 ],
         [-0.03677746, -0.03529952, -0.01952608, ..., -0.01145453,
          -0.02482045,  0.02370409],
         [ 0.01087841, -0.00978614,  0.01982535, ..., -0.0091996 ,
          -0.00344254, -0.02501079],
         ...,
         [ 0.0189324 , -0.03599298, -0.01270954, ..., -0.00581919,
          -0.02297305,  0.00045675],
         [ 0.00578905,  0.00637918,  0.02703105, ...,  0.01235018,
          -0.03318778, -0.01806359],
         [-0.01088002,  0.00967584,  0.00729706, ..., -0.03794765,
           0.0085461 ,  0.03949658]],

        [[ 0.01388897,  0.01269808, -0.01946583, ..., -0.01798273,
           0.01061713, -0.00473653],
         [ 0.03926257, -0.01340395, -0.03068396, ..., -0.01959637,
          -0.04114548,  0.02089264],
         [-0.03995167, -0.00882332, -0.00724239, ...,  0.00240645,
           0.03593447,  0.02601079],
         ...,
         [ 0.03420245,  0.02774493,  0.03424294, ..., -0.03003263,
           0.00388588,  0.01565982],
         [ 0.03053245,  0.00463836,  0.01330796, ..., -0.00890787,
          -0.02148253,  0.03047802],
         [-0.03531924, -0.03699651, -0.00251274, ..., -0.01914163,
          -0.00644141, -0.03093452]]],


       [[[ 0.01794977,  0.02249349,  0.01887948, ...,  0.0047006 ,
           0.04123594,  0.02771773],
         [ 0.02362907, -0.0352267 ,  0.01576691, ...,  0.01826527,
           0.00264788, -0.01299785],
         [-0.01435755, -0.03679209, -0.02971683, ..., -0.02388294,
           0.00520126,  0.0289862 ],
         ...,
         [-0.0080753 , -0.00530385, -0.03374789, ..., -0.02816867,
          -0.02775025,  0.01560079],
         [-0.04073375,  0.00069596, -0.00100181, ..., -0.03711006,
           0.01437995,  0.02548057],
         [ 0.04041447,  0.023376  , -0.03329135, ...,  0.02837447,
           0.02146133,  0.03326521]],

        [[-0.03946926,  0.01398532,  0.01044556, ...,  0.03271553,
           0.02026066,  0.03028153],
         [ 0.01336658,  0.01860374, -0.00655244, ..., -0.00069497,
          -0.00924619,  0.0120523 ],
         [ 0.03496051, -0.01069393, -0.00866006, ..., -0.00745103,
          -0.0075974 , -0.00659687],
         ...,
         [-0.02678686, -0.03737042, -0.01005015, ...,  0.01386445,
           0.01957164,  0.03709314],
         [-0.01638584, -0.00788457, -0.01531474, ..., -0.02616107,
          -0.00799394,  0.01629503],
         [ 0.0372932 ,  0.00273477, -0.03549632, ...,  0.04110647,
          -0.00792312,  0.02743066]],

        [[-0.01229433,  0.00834851, -0.0191376 , ...,  0.01850352,
          -0.00074403,  0.03484605],
         [ 0.00145109,  0.00816129,  0.01672349, ..., -0.0287937 ,
          -0.03861003,  0.02142001],
         [-0.03940784, -0.00510073,  0.03842596, ...,  0.01192383,
          -0.02951004,  0.02805463],
         ...,
         [-0.01643088, -0.03855009, -0.03228399, ..., -0.0389844 ,
           0.00484733, -0.03961067],
         [ 0.02934922, -0.01442853,  0.01367689, ...,  0.00288157,
           0.03880434,  0.03524129],
         [ 0.00554633,  0.03371095,  0.03511119, ..., -0.00710847,
           0.03413248, -0.0231077 ]]],


       [[[ 0.00585477,  0.02479555, -0.02526895, ...,  0.03711246,
           0.03667114, -0.00203006],
         [-0.00389084,  0.00810253,  0.03730289, ..., -0.04118606,
           0.02305565, -0.03171007],
         [ 0.0160502 , -0.00267783, -0.02397173, ...,  0.02666027,
           0.02450522,  0.00718132],
         ...,
         [ 0.0259396 , -0.01343587,  0.00787202, ..., -0.03656501,
          -0.01386203,  0.02025493],
         [-0.02964015, -0.01639891, -0.00348121, ..., -0.01103204,
           0.01647304, -0.03021587],
         [ 0.02711482,  0.00934371, -0.03973458, ..., -0.0291305
```

```
              [ 0.02711482,  0.00934371, -0.03973438, ..., -0.0291305 ,
```

Skip 6th layer

```
              [ 0.03698293,  0.01893531, -0.02712354, ..., -0.01259666,
```

```
# removing 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: 5.8056 - accuracy:
test loss, test acc: [5.797648906707764, 0.09994000196456909]
Evaluate on test data
20/20 [==============================] - 1s 44ms/step - loss: 5.8206 - accuracy:
test loss, test acc: [5.820577144622803, 0.09809999912977219]
```

Skpi 7th Layer

```
# removing 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.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: 6.3101 - accuracy:
test loss, test acc: [6.314579963684082, 0.12707999348640442]
Evaluate on test data
20/20 [==============================] - 1s 44ms/step - loss: 6.3020 - accuracy:
test loss, test acc: [6.302042007446289, 0.12470000237226486]
```

Skip 8th layer

```
# removing 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)
```

```
      1 model = eight()
----> 2 model.load_weights('Weights/MemorizeCIFAR10VGG16.h5', by_name=True)
      3 
      4 # Evaluate the model on the test data using `evaluate`
      5 print("Evaluate on train data")
```

1 frames

```
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/keras/saving/hdf5_format.py in
load_weights_from_hdf5_group_by_name(f, layers, skip_mismatch)
    786                                  symbolic_weights[i])) +
    787                             ', but the saved weight has shape ' +
--> 788                             str(weight_values[i].shape) + '.')
    789 
    790         else:

ValueError: Layer #11 (named "block4_conv2"), weight <tf.Variable
'block4_conv2/kernel:0' shape=(3, 3, 256, 512) dtype=float32, numpy=
array([[[[-1.86594147e-02, -3.68393026e-03, -1.77041981e-02, ...,
           1.79787371e-02, -1.48948580e-02, -1.11245140e-02],
         [ 2.67421063e-02, -1.64897721e-02,  7.91633688e-03, ...,
          -1.12397857e-02,  4.88701276e-03, -2.09583212e-02],
         [-2.24106014e-03,  1.22879818e-03, -3.15737538e-03, ...,
          -2.90314723e-02, -2.30356343e-02,  1.49141680e-02],
         ...,
         [-2.03294829e-02,  2.65758764e-02,  2.41820905e-02, ...,
           2.63535623e-02,  2.70044710e-02,  1.73983034e-02],
         [-1.13980249e-02,  6.24835305e-03,  1.31087955e-02, ...,
          -1.20901614e-02,  5.02749346e-03,  2.47579236e-02],
         [ 1.35941803e-03,  2.31712852e-02, -2.42953617e-02, ...,
          -5.26320748e-03,  2.69606523e-03, -1.52463764e-02]],

        [[-3.12371179e-04,  7.94581883e-03, -2.53573004e-02, ...,
           2.34961454e-02, -2.70950012e-02,  1.84722971e-02],
         [-8.48257355e-03,  3.99256311e-03,  6.97694533e-03, ...,
          -2.87482329e-03,  5.72860241e-04,  1.03627164e-02],
         [ 3.21386568e-03, -7.70348124e-03,  1.85464974e-02, ...,
           2.36590076e-02,  6.21172227e-03,  1.19324345e-02],
         ...,
         [ 2.16104183e-02,  1.35596115e-02,  1.75614264e-02, ...,
           1.64795648e-02, -2.71353573e-02, -2.16093063e-02],
         [-6.90499321e-04, -1.22751649e-02,  5.66814654e-03, ...,
           2.12768652e-03, -5.71485981e-03, -2.91666873e-02],
         [ 2.67655756e-02,  9.32419486e-03, -1.61860269e-02, ...,
          -4.28149477e-04, -1.11879166e-02,  2.50900164e-04]],

        [[-9.62005369e-03,  1.23251509e-02, -1.56112984e-02, ...,
          -2.88242232e-02,  1.27051454e-02, -1.46362744e-03],
         [ 2.11083330e-03, -1.25205778e-02, -1.59069896e-04, ...,
          -7.07139820e-03, -2.18274184e-02,  2.49787606e-03],
         [ 1.31653268e-02,  4.88095544e-03,  6.74160756e-03, ...,
           1.91649813e-02, -5.86929359e-03,  5.71096875e-03],
         ...,
         [ 9.21543688e-05,  8.65946896e-03,  8.96252878e-03, ...,
          -1.55703733e-02,  2.77267415e-02, -2.22417414e-02],
         [ 2.00977828e-02,  2.06327941e-02,  1.90037545e-02, ...,
          -2.05035713e-02, -2.63002384e-02, -2.53258273e-03],
         [-2.61365473e-02,  9.55066644e-03, -2.57531572e-02, ...,
           1.82334017e-02,  1.49432570e-04, -1.17539912e-02]]],


       [[[-9.56375152e-04, -9.85068083e-03, -8.06411169e-03, ...,
           5.08235581e-03,  3.13576311e-05,  1.41604263e-02],
         [-1.49032520e-02, -1.31982286e-02, -1.16564147e-03, ...,
          -4.71525639e-03,  1.60135049e-02,  2.10340042e-02],
         [-7.73567520e-03, -6.19293936e-03, -7.71755166e-03, ...,
          -2.00744458e-02,  6.55433349e-03,  5.13862260e-03],
         ...,
         [-2.82618850e-02,  2.44522672e-02,  2.21129339e-02, ...,
          -2.12434381e-02,  2.66612601e-02,  8.11996125e-03],
         [-4.22681682e-03,  1.18083674e-02, -1.82898920e-02, ...,
          -1.28730461e-02, -3.97273339e-03,  1.09876189e-02],
         [-1.99060328e-03,  2.54793521e-02, -2.15872638e-02, ...,
          -2.10956074e-02,  1.62722301e-02,  3.00797261e-03]],

        [[ 7.97907822e-03,  1.27929170e-02, -1.34360213e-02, ...,
          -1.92722995e-02, -2.65571009e-02, -1.93730183e-03],
         [-2.62388159e-02, -1.42331235e-03,  2.21080240e-02, ...,
           1.90000813e-02, -2.32438613e-02,  2.28278767e-02],
         [ 2.49010045e-02,  2.78347638e-02,  2.44759042e-02, ...,
          -1.77527796e-02, -1.61872357e-02,  5.85570000e-03],
         ...,
         [-2.29183808e-02,  2.24572588e-02,  2.43437756e-02, ...,
          -1.77322682e-02,  1.07564796e-02,  4.48325090e-03],
         [-2.35843994e-02, -1.27329770e-02, -1.94379725e-02, ...,
          -2.93060038e-02,  1.05252992e-02, -2.58581731e-02],
         [-1.75081380e-02,  2.17381772e-02,  2.60184091e-02, ...,
          -1.14804581e-02, -2.04935540e-02,  1.74890514e-02]],

        [[ 2.54785437e-02, -2.44980659e-02, -4.87575866e-03, ...,
           2.00088713e-02,  1.65627915e-02,  2.13502292e-02],
         [-1.93291903e-02, -2.47890688e-02, -3.50005738e-03, ...,
          -1.03630666e-02, -1.16968956e-02, -3.36969644e-03],
         [ 8.96027312e-04, -4.96399961e-03,  2.92887930e-02, ...,
           2.30720453e-03,  1.56256128e-02, -1.31672733e-02],
         ...,
         [ 1.26650557e-03,  4.77652438e-03,  2.79479977e-02, ...,
          -4.04876657e-03, -1.99722759e-02, -6.49709068e-03],
         [-2.08868906e-02,  8.13687220e-04,  1.03679206e-02, ...,
          -2.57654358e-02, -1.06997788e-02, -1.85028799e-02],
         [ 2.73699500e-03,  7.63737597e-03, -2.82778442e-02, ...,
          -1.53870508e-03,  9.25524160e-04,  2.68824752e-02]]],


       [[[ 1.31814610e-02,  2.35662479e-02, -4.57051024e-03, ...,
          -3.01882438e-03, -1.55177107e-02, -1.15938894e-02],
         [-1.34573281e-02,  2.00580005e-02,  1.41060445e-02, ...,
```

Skip 9th layer

```
          [ 1.38650332e-02, -2.40237806e-02,  1.95958596e-02,
```

```
# removing 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)
```

```
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: 15.2275 - accuracy
    test loss, test acc: [15.269312858581543, 0.08243999630212784]
    Evaluate on test data
    20/20 [==============================] - 1s 42ms/step - loss: 15.2717 - accuracy
    test loss, test acc: [15.27167797088623, 0.0851999968290329]
```

Skip 10th layer

```
# removing 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'))
    # 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 44ms/step - loss: 31.8018 - accuracy
test loss, test acc: [31.84032440185547, 0.11209999769926071]
Evaluate on test data
20/20 [==============================] - 1s 43ms/step - loss: 32.1203 - accuracy
test loss, test acc: [32.12032699584961, 0.10679999738931656]
```

Skip 11th layer

```
# removing 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'))
```