112-1 PR for UTA/NTUT EMBA 2023/9/17

111C51501
111C51502
111C51523

- Cifar10 testing accuracy = 59.52%
- Cifar100 testing accuracy = 32.92%

對於相同的模型和訓練策略，cifar10 的準確率會比 cifar100 高。原因如下：

- 類別數量：cifar10 只有 10 個類別，而 cifar100 有 100 個類別。當類別數量增加時，分類任務通常會變得更加困難，因為模型需要學習更多的特徵來區分更多的類別。

- 類別間的差異：在 cifar10 中，類別之間的差異相對較大（例如，飛機和貓）。但在 cifar100 中，由於有更多的類別，某些類別之間的差異可能會較小，這使得分類更加困難。

- 數據量：雖然兩個數據集的總圖像數量都是 60,000 張，但對於每個類別，cifar10 提供了 5,000 張訓練圖像，而 cifar100 只提供了 500 張。更多的訓練數據通常意味著更好的模型性能。

- 挑戰性：由於 cifar100 的類別更多且更細，它通常被認為是一個更具挑戰性的數據集。

總之，如果使用相同的模型和訓練策略，您應該期望在 cifar10 上獲得更高的準確率。然而，這不意味著 cifar10 是一個更好的數據集，只是它相對較簡單。

使用 CIFAR100，10 個分類

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
print('tensorflow', tf.__version__)
```

```
tensorflow 2.13.0
```

```
model_VGG16 = tf.keras.applications.VGG16(include_top=True, weights='imagenet')
model_VGG16.summary()
```

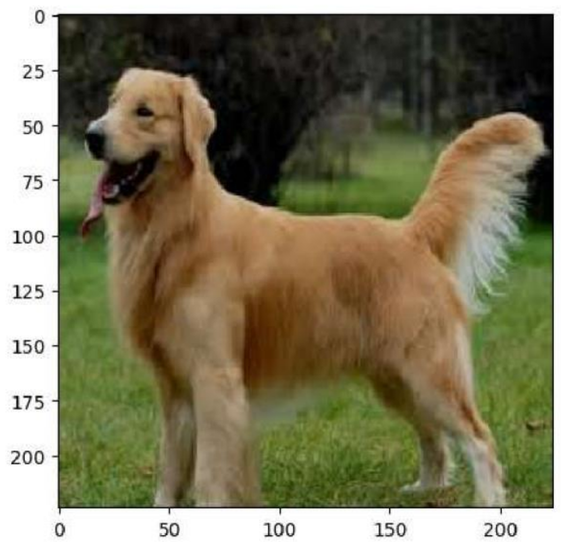
```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 predictions (Dense)         (None, 1000)              4097000

=================================================================
Total params: 138357544 (527.79 MB)
Trainable params: 138357544 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model_VGG16_notop = tf.keras.applications.VGG16(include_top=False, weights='imagenet', input_shape = (32,32,3))
model_VGG16_notop.summary()
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 32, 32, 3)]       0

 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

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
img = tf.keras.preprocessing.image.load_img('/content/gold_dog.jpg',target_size=(224,224))
# img = tf.keras.preprocessing.image.load_img('/content/gold_dog.jpg',target_size=(32,32))
img = np.array(img)
plt.imshow(img)
print(img.shape)
print('R channel AVG:', np.mean(img[:,:,0]))
print('G channel AVG:', np.mean(img[:,:,1]))
print('B channel AVG:', np.mean(img[:,:,2]))
```

```
(224, 224, 3)
R channel AVG: 96.29793128188776
G channel AVG: 90.23985570790816
B channel AVG: 54.940788424744895
```



```
x = np.expand_dims(img, axis=0)
x = tf.keras.applications.vgg16.preprocess_input(x)
print(x.shape)
print('R channel AVG:', np.mean(x[0,:,:,0]))
print('G channel AVG:', np.mean(x[0,:,:,1]))
print('B channel AVG:', np.mean(x[0,:,:,2]))
```

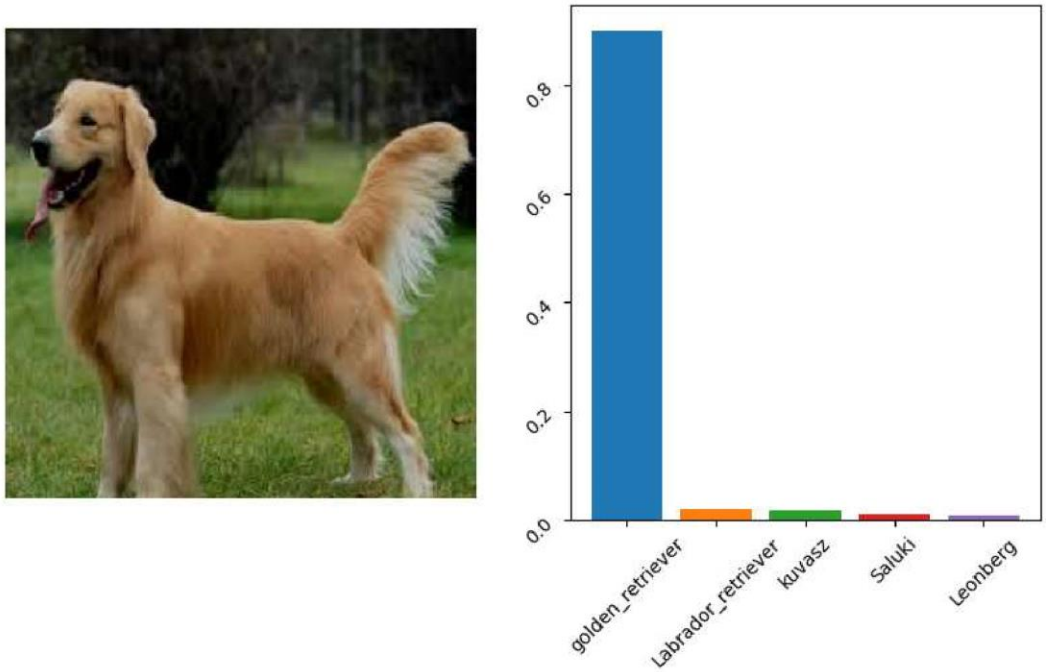
```
(1, 224, 224, 3)
R channel AVG: -48.998215
G channel AVG: -26.539143
B channel AVG: -27.382067
```

```
y_pred = model_VGG16.predict(x)
top_prediction = tf.keras.applications.vgg16.decode_predictions(y_pred, top=5)[0]
top_prediction
```

```
1/1 [==============================] - 1s 1s/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 [==============================] - 0s 0us/step
[('n02099601', 'golden_retriever', 0.9001082),
 ('n02099712', 'Labrador_retriever', 0.021917846),
 ('n02104029', 'kuvasz', 0.018392105),
 ('n02091831', 'Saluki', 0.010036397),
 ('n02111129', 'Leonberg', 0.007826946)]
```

```
fig, (ax1,ax2) = plt.subplots(1, 2, figsize=(10, 5), dpi=100)
ax1.imshow(img)
ax1.set_axis_off()
for k, (class_name, class_description, score) in enumerate(top_prediction):
    print(f'top-{k + 1} is {class_description}. ({score:.2f})')
    ax2.bar(class_description, score)
    ax2.tick_params (labelrotation=45)
```

```
top-1 is golden_retriever. (0.90)
top-2 is Labrador_retriever. (0.02)
top-3 is kuvasz. (0.02)
top-4 is Saluki. (0.01)
top-5 is Leonberg. (0.01)
```

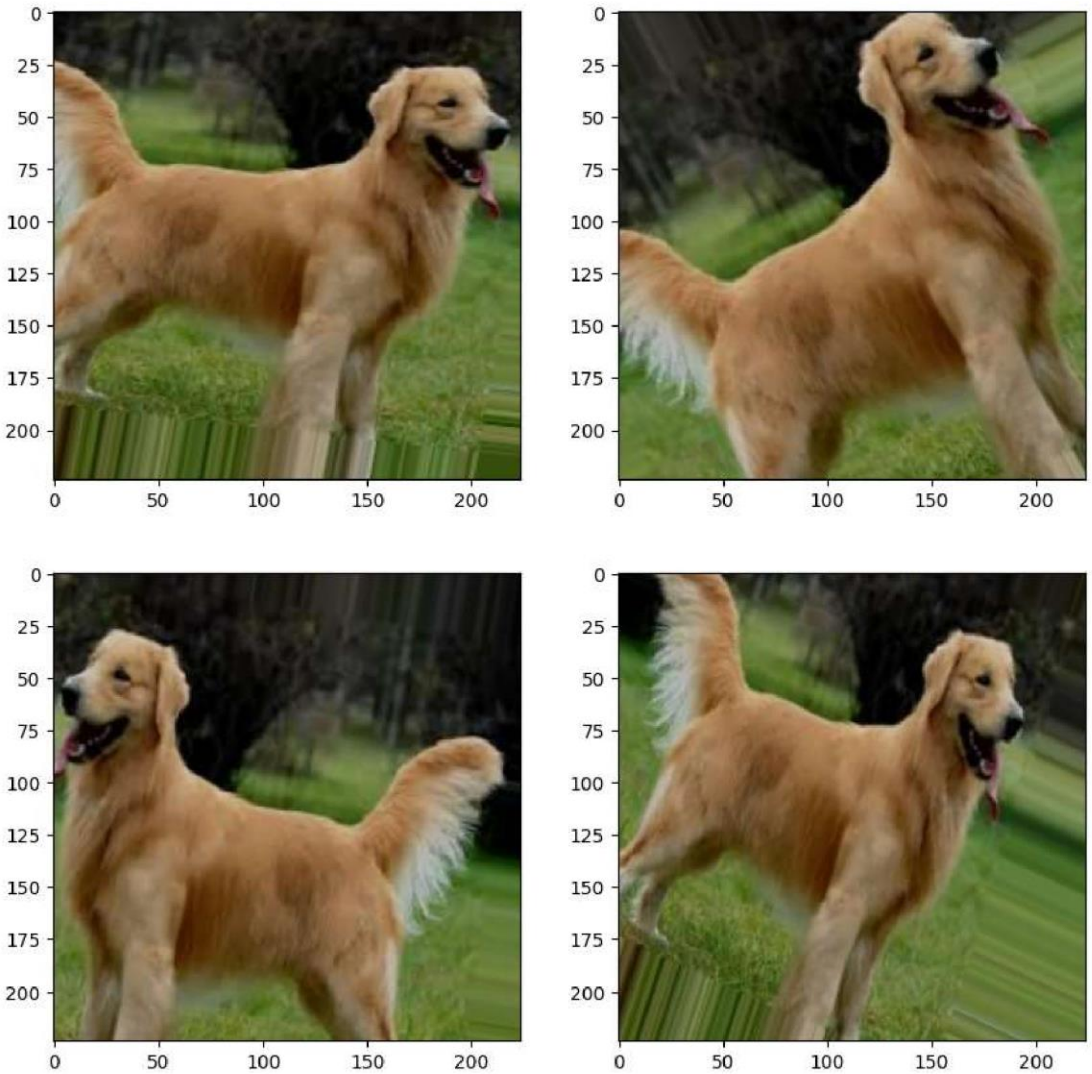



```
datagen = ImageDataGenerator(
    # rescale=1./255,
    rotation_range=30,          # 隨機旋轉的度數範圍。
    width_shift_range=0.1,      # 水平位置平移 距離上限為 寬度乘以參數
    height_shift_range=0.1,   # 垂直位置平移 距離上限為 寬度乘以參數
    shear_range=0.2,            #剪切強度
    zoom_range=0.2,           # 圖片縮放<1 為放大 >1 為縮小
    # channel_shift_range=0.0,  # 通道數量偏移 用來改變圖片顏色
    horizontal_flip=True,     #隨機水平翻轉
    fill_mode='nearest',
    validation_split=0.2
    # 所有參數說明 https://keras.io/zh/preprocessing/image/
)
```

```
img = tf.keras.preprocessing.image.load_img('/content/gold_dog.jpg',target_size=(224,224))
m = tf.keras.preprocessing.image.img_to_array(img)
print(m.shape)
n = []
n.append(m)
n = np.array(n)
print(n.shape)
```

```
(224, 224, 3)
(1, 224, 224, 3)
```

```
# 產生資料增強圖片
augmented_images = datagen.flow(n, batch_size=1)

# 隨機取得4 張資料增強圖片並顯示
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 10), dpi=100)
axes = axes.flatten()

for i in range(4):
    # 取得下一張資料增強圖片
    augmented_image = augmented_images.next()[0]
    augmented_image = augmented_image.astype('uint8')
    axes[i].imshow(augmented_image)

plt.show()
```




```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# 資料拆分
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size = 0.2)

# one-hot encodering
y_train = to_categorical(y_train, num_classes = 10)
y_val = to_categorical(y_val, num_classes = 10)
y_test = to_categorical(y_test, num_classes = 10)
```

```
#  資料前處理
x_train  =  x_train  *  1.0/255
x_val  =  x_val  *  1.0/255
x_test  =  x_test  *  1.0/255


print(x_train.shape,  x_val.shape,  x_test.shape)
print(y_train.shape,  y_val.shape,  y_test.shape)
```

```
(40000, 32, 32, 3) (10000, 32, 32, 3) (10000, 32, 32, 3)
(40000, 10) (10000, 10) (10000, 10)
```

```
train_datagen  =  ImageDataGenerator(
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip=True
)
train_datagen.fit(x_train)


checkpoint_filepath  =  './check.h5'

callback_checkpoint  =  tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True
)

reduce_learning_rate=  tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_accuracy',
    mode='max',
    factor=0.8,
    patience=3,
    cooldown=0,
    min_lr=0.000001,
    verbose=1
)

callback_Earlystop  =  tf.keras.callbacks.EarlyStopping  (monitor=  'val_accuracy',mode='max',  patience=3)


model_VGG16_notop  =  tf.keras.applications.VGG16(include_top=False,  weights='imagenet',  input_shape=(32,32,3))

model_VGG16_notop.trainable  =  False  #  True表示參與訓練,  False表示凍結權重

x  =  model_VGG16_notop.output
x  =  Flatten()(x)
x  =  tf.keras.layers.Dense(4096,  activation='relu')(x)
x  =  tf.keras.layers.Dense(4096,  activation='relu')(x)
predictions  =  tf.keras.layers.Dense(10,activation='softmax')(x)

model  =  Model(model_VGG16_notop.input,  predictions)

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


history  =  model.fit(
        train_datagen.flow(x_train,  y_train,  batch_size=128),
        validation_data  =  (x_val,  y_val),
        epochs  =  3,
        verbose  =  1,
        callbacks  =  [callback_checkpoint,  reduce_learning_rate,  callback_Earlystop]
)


loss,  accuracy  =  model.evaluate(x_test,  y_test,  verbose=0)
print("Testing  Accuracy  =  %.2f  %%        loss  =  %f"  %  (accuracy*100,  loss))
```

```
Epoch 1/3
313/313 [==============================] - 904s 3s/step - loss: 1.4702 - accuracy: 0.4869 - val_loss: 1.2098 - val_accuracy: 0.5748 - lr: 0.0010
Epoch 2/3
313/313 [==============================] - 901s 3s/step - loss: 1.2459 - accuracy: 0.5616 - val_loss: 1.1800 - val_accuracy: 0.5882 - lr: 0.0010
Epoch 3/3
313/313 [==============================] - 901s 3s/step - loss: 1.1870 - accuracy: 0.5807 - val_loss: 1.1414 - val_accuracy: 0.6035 - lr: 0.0010
Testing Accuracy = 59.52 %    loss = 1.156711
```

使用 CIFAR100，100 個分類

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import cifar10, cifar100
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
print('tensorflow', tf.__version__)
```

```
tensorflow 2.13.0
```

```
model_VGG16 = tf.keras.applications.VGG16(include_top=True, weights='imagenet')
model_VGG16.summary()
```

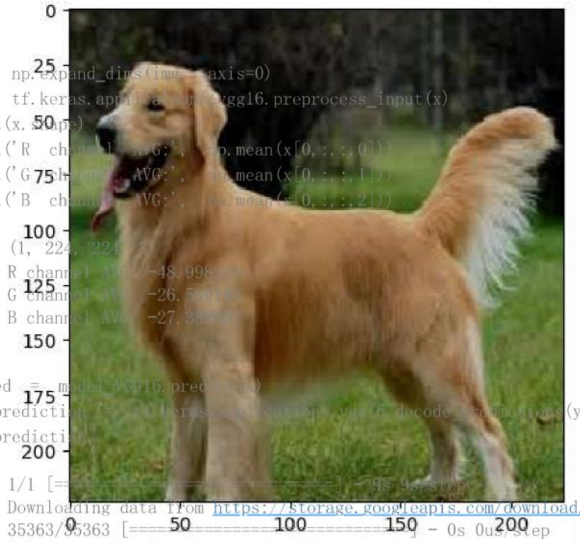
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 [==============================] - 25s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 predictions (Dense)         (None, 1000)              4097000

=================================================================
Total params: 138357544 (527.79 MB)
Trainable params: 138357544 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model_VGG16_notop = tf.keras.applications.VGG16(include_top=False, weights='imagenet', input_shape = (32,32,3))
model_VGG16_notop.summary()
# (x_train, y_train), (x_test, y_test) = cifar10.load_data()
(x_train, y_train), (x_test, y_test) = cifar100.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 4s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 32, 32, 3)]       0

 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

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz
169001437/169001437 [==============================] - 14s 0us/step
```

```
img = tf.keras.preprocessing.image.load_img('/content/gold_dog.jpg',target_size=(224,224))
# img = tf.keras.preprocessing.image.load_img('/content/gold_dog.jpg',target_size=(32,32))
img = np.array(img)
plt.imshow(img)
print(img.shape)
print('R channel AVG:', np.mean(img[:,:,0]))
print('G channel AVG:', np.mean(img[:,:,1]))
print('B channel AVG:', np.mean(img[:,:,2]))
```

```
(224, 224, 3)
R channel AVG: 96.29793128188776
G channel AVG: 90.23985570790816
B channel AVG: 54.940788424744895
```



```
x = np.expand_dims(img, axis=0)
x = tf.keras.applications.vgg16.preprocess_input(x)
print(x.shape)
print('R channel AVG:', np.mean(x[0,:,:,0]))
print('G channel AVG:', np.mean(x[0,:,:,1]))
print('B channel AVG:', np.mean(x[0,:,:,2]))
```

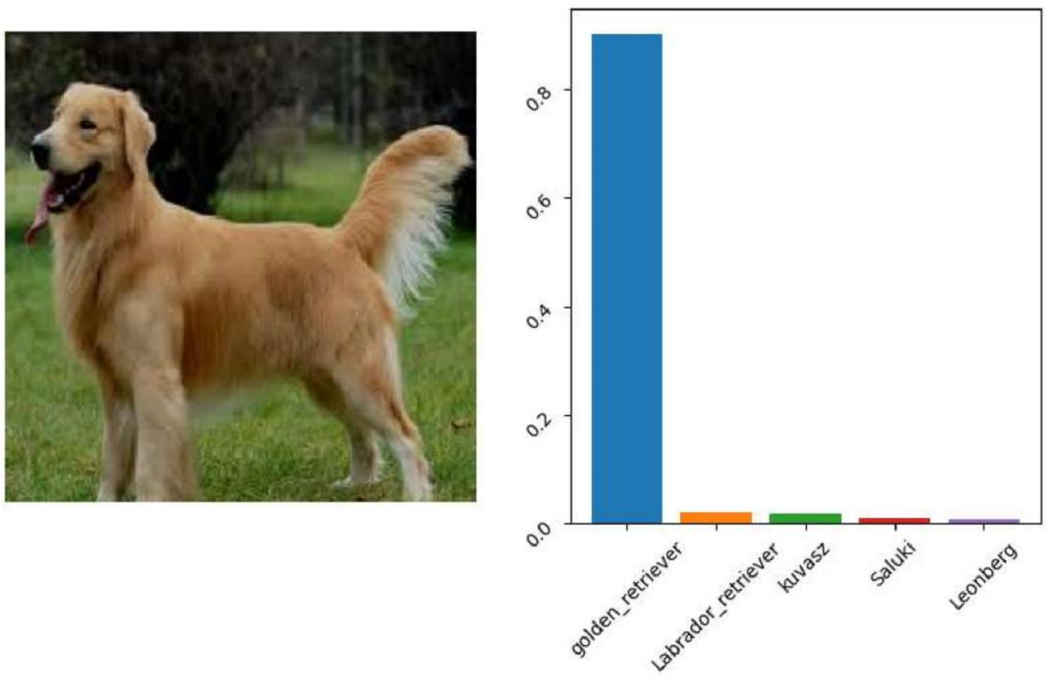
```
(1, 224, 224
R channel AVG: -48.998
G channel AVG: -26.5
B channel AVG: -27.38
```

```
y_pred = model_VGG16.predict(x)
top_prediction = tf.keras.applications.vgg16.decode_predictions(y_pred, top=5)[0]
top_prediction
```

```
1/1 [==============================] - 9s 9s/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 [==============================] - 0s 0us/step
[('n02099601', 'golden_retriever', 0.9001086),
 ('n02099712', 'Labrador_retriever', 0.021917732),
 ('n02104029', 'kuvasz', 0.018392079),
 ('n02091831', 'Saluki', 0.010036401),
 ('n02111129', 'Leonberg', 0.007826927)]
```

```
fig, (ax1,ax2) = plt.subplots(1, 2, figsize=(10, 5), dpi=100)
ax1.imshow(img)
ax1.set_axis_off()
for k, (class_name, class_description, score) in enumerate(top_prediction):
    print(f'top-{k + 1} is {class_description}. ({score:.2f})')
    ax2.bar(class_description, score)
    ax2.tick_params (labelrotation=45)
```

```
top-1 is golden_retriever. (0.90)
top-2 is Labrador_retriever. (0.02)
top-3 is kuvasz. (0.02)
top-4 is Saluki. (0.01)
top-5 is Leonberg. (0.01)
```



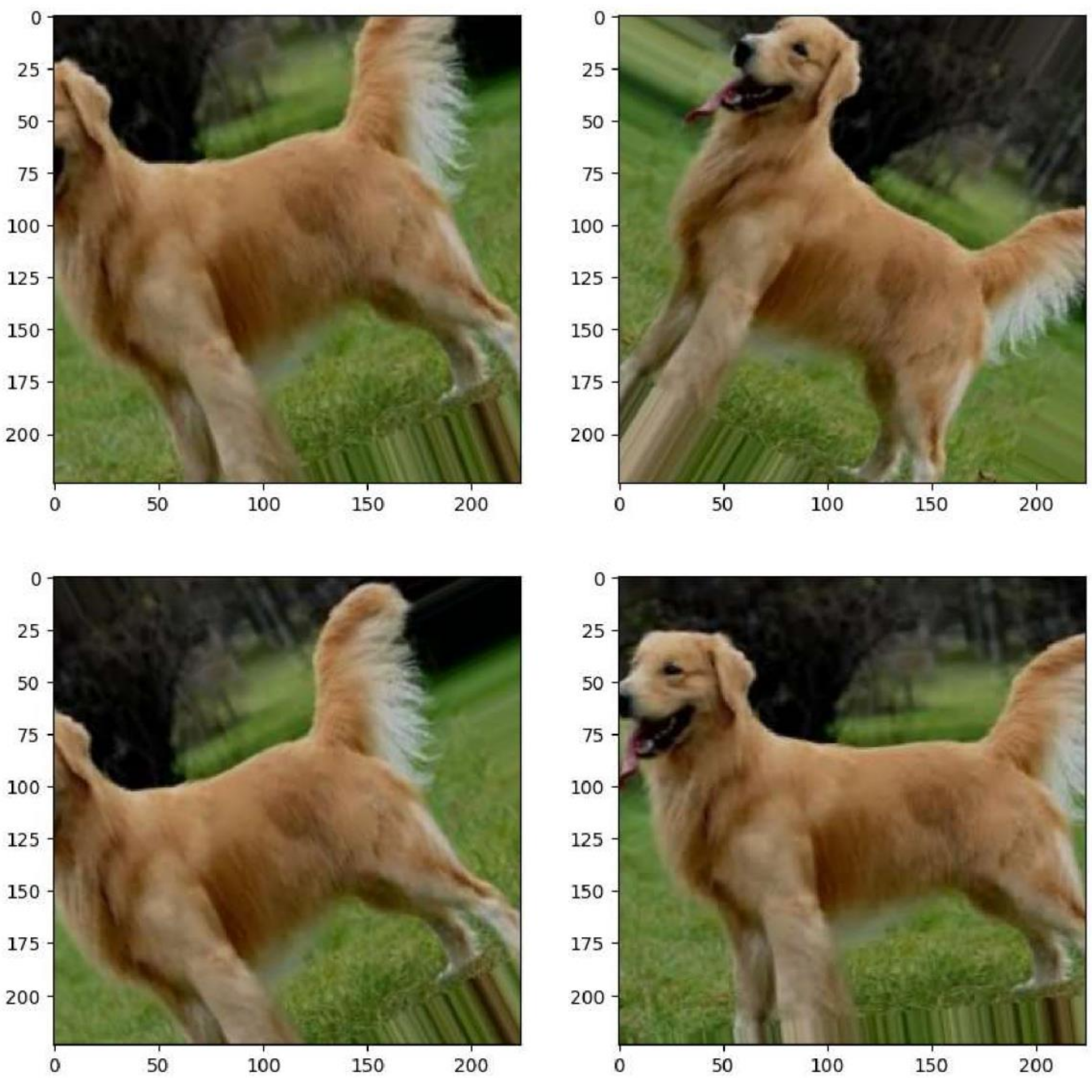

```
datagen = ImageDataGenerator(
    # rescale=1./255,
    rotation_range=30,          # 隨機旋轉的度數範圍。
    width_shift_range=0.1,      # 水平位置平移 距離上限為 寬度乘以參數
    height_shift_range=0.1,    # 垂直位置平移 距離上限為 寬度乘以參數
    shear_range=0.2,            #剪切强度
    zoom_range=0.2,            # 圖片縮放<1 為放大 >1 為縮小
    # channel_shift_range=0.0,  # 通道數量偏移 用來改變圖片顏色
    horizontal_flip=True,      #隨機水平翻轉
```

```
    fill_mode='nearest',
    validation_split=0.2
    # 所有参數説明 https://keras.io/zh/preprocessing/image/
)
```

```
img = tf.keras.preprocessing.image.load_img('/content/gold_dog.jpg',target_size=(224,224))
m = tf.keras.preprocessing.image.img_to_array(img)
print(m.shape)
n = []
n.append(m)
n = np.array(n)
print(n.shape)
```

```
(224, 224, 3)
(1, 224, 224, 3)
```

```
# 產生資料增強圖片
augmented_images = datagen.flow(n, batch_size=1)

# 隨機取得4 張資料增強圖片並顯示
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 10), dpi=100)
axes = axes.flatten()

for i in range(4):
    # 取得下一張資料增強圖片
    augmented_image = augmented_images.next()[0]
    augmented_image = augmented_image.astype('uint8')
    axes[i].imshow(augmented_image)

plt.show()
```



```
# (x_train, y_train), (x_test, y_test) = cifar10.load_data()
(x_train, y_train), (x_test, y_test) = cifar100.load_data()
```

```
# 資料拆分
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size = 0.2)


# one-hot encodering

# y_train = to_categorical(y_train, num_classes = 10)
# y_val = to_categorical(y_val, num_classes = 10)
# y_test = to_categorical(y_test, num_classes = 10)

y_train = to_categorical(y_train, num_classes = 100)
y_val = to_categorical(y_val, num_classes = 100)
y_test = to_categorical(y_test, num_classes = 100)


# 資料前處理
x_train = x_train * 1.0/255
x_val = x_val * 1.0/255
x_test = x_test * 1.0/255


print(x_train.shape, x_val.shape, x_test.shape)
print(y_train.shape, y_val.shape, y_test.shape)

    (40000, 32, 32, 3) (10000, 32, 32, 3) (10000, 32, 32, 3)
    (40000, 100) (10000, 100) (10000, 100)


train_datagen = ImageDataGenerator(
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip=True
)
train_datagen.fit(x_train)


checkpoint_filepath = './check.h5'

callback_checkpoint = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True
)

reduce_learning_rate= tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_accuracy',
    mode='max',
    factor=0.8,
    patience=3,
    cooldown=0,
    min_lr=0.000001,
    verbose=1
)

callback_Earlystop = tf.keras.callbacks.EarlyStopping (monitor= 'val_accuracy',mode='max', patience=3)


model_VGG16_notop = tf.keras.applications.VGG16(include_top=False, weights='imagenet', input_shape=(32,32,3))

model_VGG16_notop.trainable = False # True表示參與訓練, False表示凍結權重

x = model_VGG16_notop.output
x = Flatten()(x)
x = tf.keras.layers.Dense(4096, activation='relu')(x)
x = tf.keras.layers.Dense(4096, activation='relu')(x)
# predictions = tf.keras.layers.Dense(10,activation='softmax')(x)
predictions = tf.keras.layers.Dense(100,activation='softmax')(x)

model = Model(model_VGG16_notop.input, predictions)

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


history = model.fit(
        train_datagen.flow(x_train, y_train, batch_size=128),
        validation_data = (x_val, y_val),
        epochs = 3,
        verbose = 1,
        callbacks = [callback_checkpoint, reduce_learning_rate, callback_Earlystop]
)


loss, accuracy = model.evaluate(x_test, y_test, verbose=0)
```

```
print("Testing  Accuracy  =  %.2f  %%        loss  =  %f"  %  (accuracy*100,  loss))
```

```
Epoch 1/3
313/313 [==============================] - 31s 91ms/step - loss: 3.2774 - accuracy: 0.2108 - val_loss: 2.8930 - val_accuracy: 0.2770 - lr: 0.001
Epoch 2/3
313/313 [==============================] - 27s 85ms/step - loss: 2.7854 - accuracy: 0.2944 - val_loss: 2.7716 - val_accuracy: 0.2994 - lr: 0.001
Epoch 3/3
313/313 [==============================] - 26s 84ms/step - loss: 2.5884 - accuracy: 0.3348 - val_loss: 2.6874 - val_accuracy: 0.3168 - lr: 0.001
Testing Accuracy = 32.92 %    loss = 2.635050
```