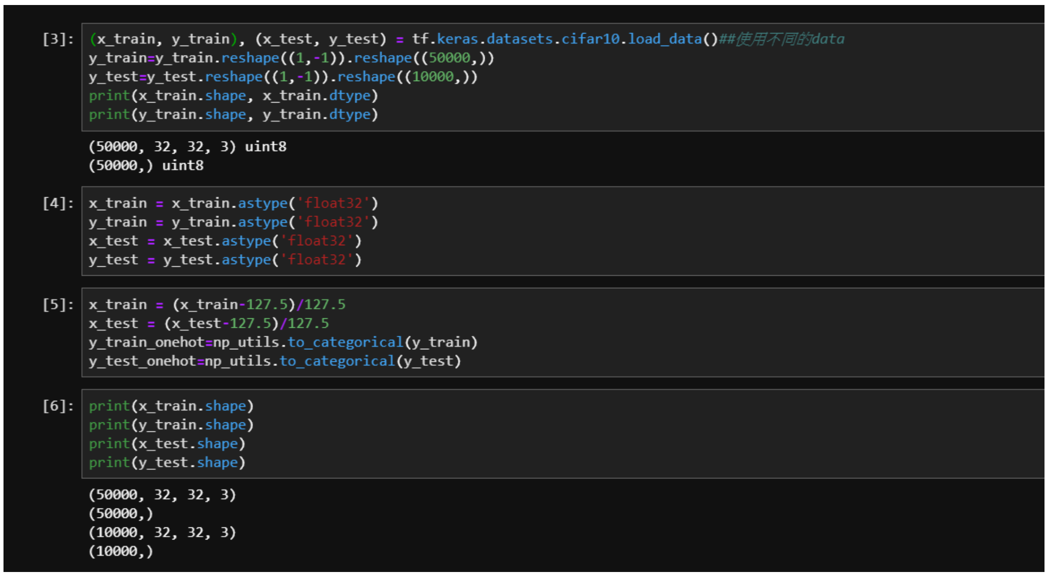
使用 cifar10 來當做分類的 dataset

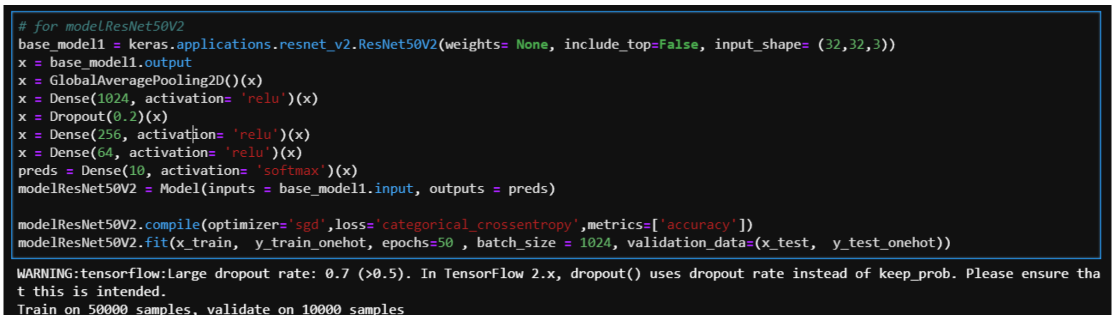
(a).framework 使用 tensorflow



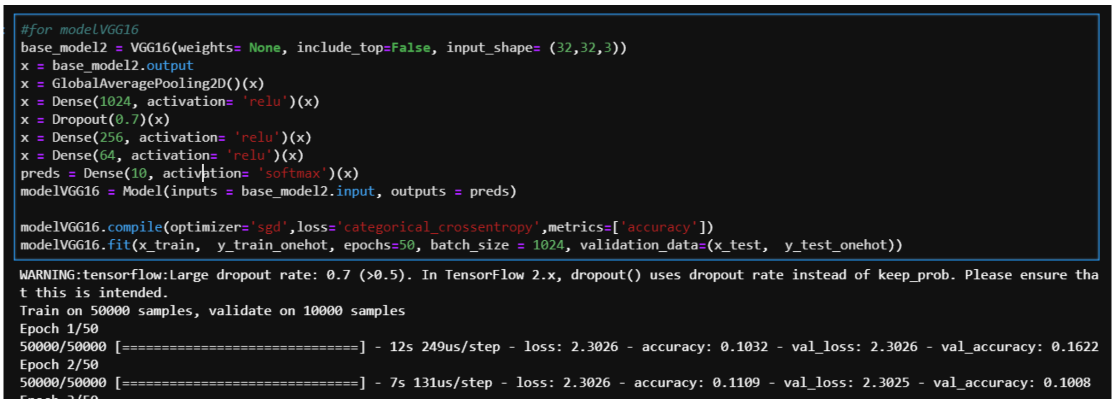
```
[3]: (x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()##使用不同的data
     y_train=y_train.reshape((1,-1)).reshape((50000,))
     y_test=y_test.reshape((1,-1)).reshape((10000,))
     print(x_train.shape, x_train.dtype)
     print(y_train.shape, y_train.dtype)

     (50000, 32, 32, 3) uint8
     (50000,) uint8

[4]: x_train = x_train.astype('float32')
     y_train = y_train.astype('float32')
     x_test = x_test.astype('float32')
     y_test = y_test.astype('float32')

[5]: x_train = (x_train-127.5)/127.5
     x_test = (x_test-127.5)/127.5
     y_train_onehot=np_utils.to_categorical(y_train)
     y_test_onehot=np_utils.to_categorical(y_test)

[6]: print(x_train.shape)
     print(y_train.shape)
     print(x_test.shape)
     print(y_test.shape)

     (50000, 32, 32, 3)
     (50000,)
     (10000, 32, 32, 3)
     (10000,)
```

使用的 dataset training 有 50000 張　testing 有 10000 張

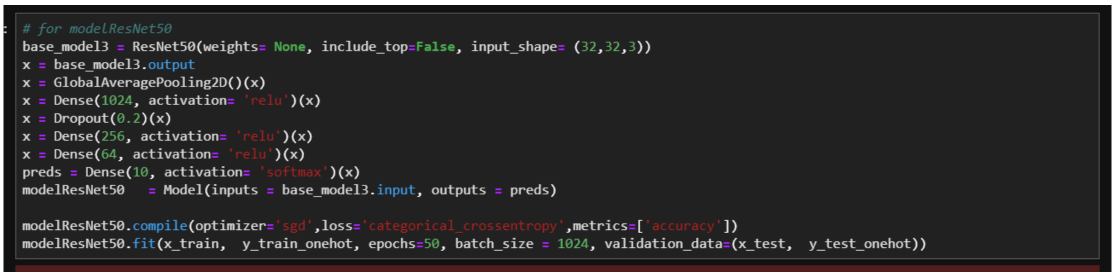
(b)使用 backbone 有 ResNet50V2, VGG16.PNG, ResNet50



```
# for modelResNet50V2
base_model1 = keras.applications.resnet_v2.ResNet50V2(weights= None, include_top=False, input_shape= (32,32,3))
x = base_model1.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation= 'relu')(x)
x = Dropout(0.2)(x)
x = Dense(256, activation= 'relu')(x)
x = Dense(64, activation= 'relu')(x)
preds = Dense(10, activation= 'softmax')(x)
modelResNet50V2 = Model(inputs = base_model1.input, outputs = preds)

modelResNet50V2.compile(optimizer='sgd',loss='categorical_crossentropy',metrics=['accuracy'])
modelResNet50V2.fit(x_train,  y_train_onehot, epochs=50 , batch_size = 1024, validation_data=(x_test,  y_test_onehot))

WARNING:tensorflow:Large dropout rate: 0.7 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate instead of keep_prob. Please ensure tha
t this is intended.
Train on 50000 samples, validate on 10000 samples
```



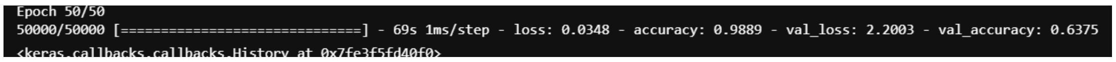
```
#for modelVGG16
base_model2 = VGG16(weights= None, include_top=False, input_shape= (32,32,3))
x = base_model2.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation= 'relu')(x)
x = Dropout(0.7)(x)
x = Dense(256, activation= 'relu')(x)
x = Dense(64, activation= 'relu')(x)
preds = Dense(10, activation= 'softmax')(x)
modelVGG16 = Model(inputs = base_model2.input, outputs = preds)

modelVGG16.compile(optimizer='sgd',loss='categorical_crossentropy',metrics=['accuracy'])
modelVGG16.fit(x_train,  y_train_onehot, epochs=50, batch_size = 1024, validation_data=(x_test,  y_test_onehot))

WARNING:tensorflow:Large dropout rate: 0.7 (>0.5). In TensorFlow 2.x, dropout() uses dropout rate instead of keep_prob. Please ensure tha
t this is intended.
Train on 50000 samples, validate on 10000 samples
Epoch 1/50
50000/50000 [==============================] - 12s 249us/step - loss: 2.3026 - accuracy: 0.1032 - val_loss: 2.3026 - val_accuracy: 0.1622
Epoch 2/50
50000/50000 [==============================] - 7s 131us/step - loss: 2.3026 - accuracy: 0.1109 - val_loss: 2.3025 - val_accuracy: 0.1008
```



```
# for modelResNet50
base_model3 = ResNet50(weights= None, include_top=False, input_shape= (32,32,3))
x = base_model3.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation= 'relu')(x)
x = Dropout(0.2)(x)
x = Dense(256, activation= 'relu')(x)
x = Dense(64, activation= 'relu')(x)
preds = Dense(10, activation= 'softmax')(x)
modelResNet50    = Model(inputs = base_model3.input, outputs = preds)

modelResNet50.compile(optimizer='sgd',loss='categorical_crossentropy',metrics=['accuracy'])
modelResNet50.fit(x_train,  y_train_onehot, epochs=50, batch_size = 1024, validation_data=(x_test,  y_test_onehot))
```

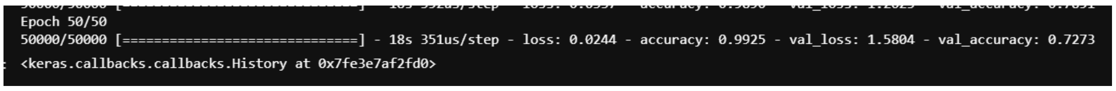
使用成效,以訓練 50epoch 比較

ResNet50V2 大約 63.75%



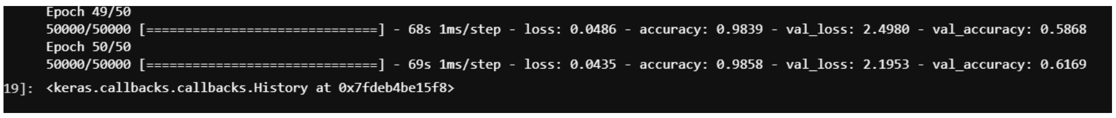
```
Epoch 50/50
50000/50000 [==============================] - 69s 1ms/step - loss: 0.0348 - accuracy: 0.9889 - val_loss: 2.2003 - val_accuracy: 0.6375
<keras.callbacks.callbacks.History at 0x7fe3f5fd40f0>
```

VGG16 大約 72.72%



```
 Epoch 50/50
 50000/50000 [==============================] - 18s 351us/step - loss: 0.0244 - accuracy: 0.9925 - val_loss: 1.5804 - val_accuracy: 0.7273
: <keras.callbacks.callbacks.History at 0x7fe3e7af2fd0>
```

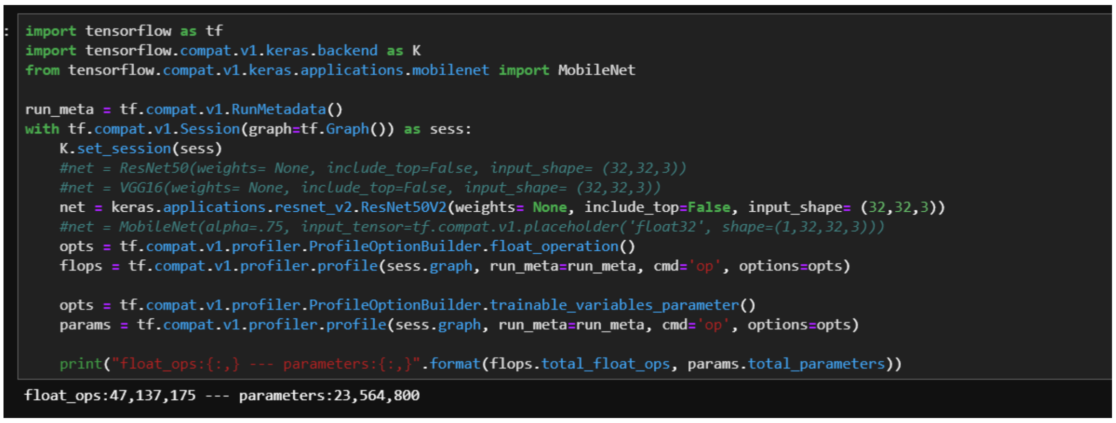
ResNet50 大約 61.69%



```
     Epoch 49/50
     50000/50000 [==============================] - 68s 1ms/step - loss: 0.0486 - accuracy: 0.9839 - val_loss: 2.4980 - val_accuracy: 0.5868
     Epoch 50/50
     50000/50000 [==============================] - 69s 1ms/step - loss: 0.0435 - accuracy: 0.9858 - val_loss: 2.1953 - val_accuracy: 0.6169
19]: <keras.callbacks.callbacks.History at 0x7fdeb4be15f8>
```

(3)比較修改過後的 flops

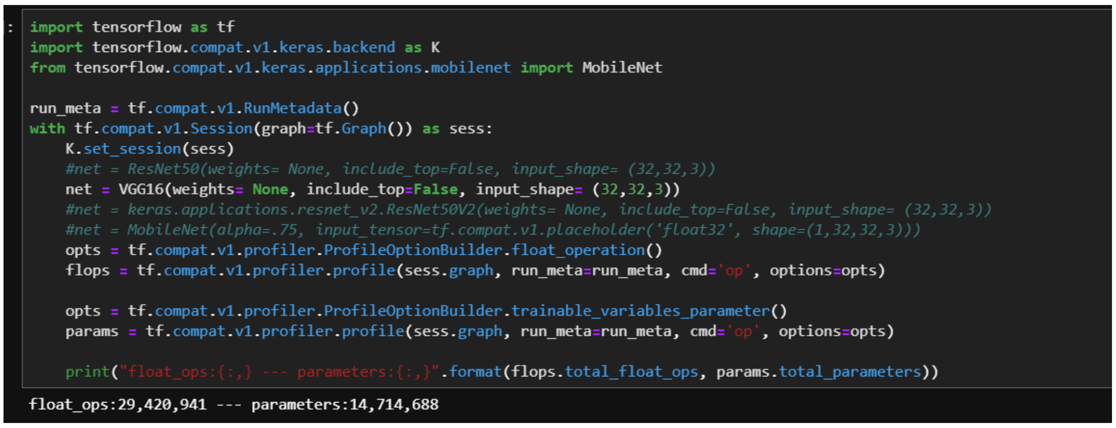
ResNet50V2



```
import tensorflow as tf
import tensorflow.compat.v1.keras.backend as K
from tensorflow.compat.v1.keras.applications.mobilenet import MobileNet

run_meta = tf.compat.v1.RunMetadata()
with tf.compat.v1.Session(graph=tf.Graph()) as sess:
    K.set_session(sess)
    #net = ResNet50(weights= None, include_top=False, input_shape= (32,32,3))
    #net = VGG16(weights= None, include_top=False, input_shape= (32,32,3))
    net = keras.applications.resnet_v2.ResNet50V2(weights= None, include_top=False, input_shape= (32,32,3))
    #net = MobileNet(alpha=.75, input_tensor=tf.compat.v1.placeholder('float32', shape=(1,32,32,3)))
    opts = tf.compat.v1.profiler.ProfileOptionBuilder.float_operation()
    flops = tf.compat.v1.profiler.profile(sess.graph, run_meta=run_meta, cmd='op', options=opts)

    opts = tf.compat.v1.profiler.ProfileOptionBuilder.trainable_variables_parameter()
    params = tf.compat.v1.profiler.profile(sess.graph, run_meta=run_meta, cmd='op', options=opts)

    print("float_ops:{:,} --- parameters:{:,}".format(flops.total_float_ops, params.total_parameters))
```

```
float_ops:47,137,175 --- parameters:23,564,800
```

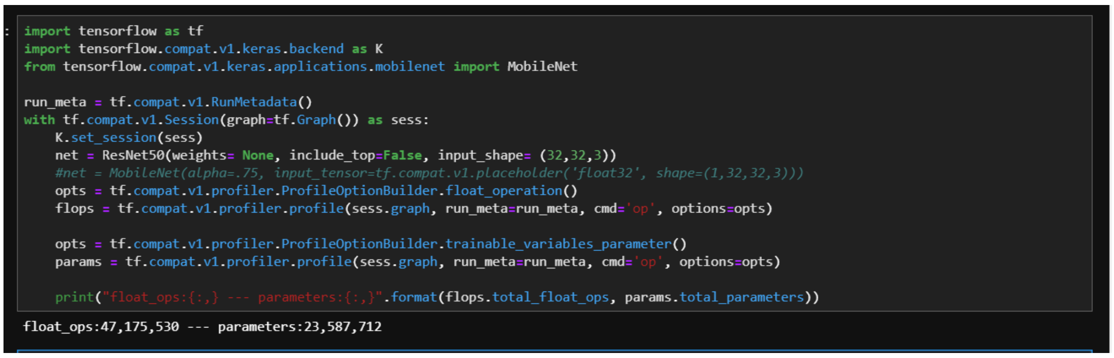
VGG16



```
import tensorflow as tf
import tensorflow.compat.v1.keras.backend as K
from tensorflow.compat.v1.keras.applications.mobilenet import MobileNet

run_meta = tf.compat.v1.RunMetadata()
with tf.compat.v1.Session(graph=tf.Graph()) as sess:
    K.set_session(sess)
    #net = ResNet50(weights= None, include_top=False, input_shape= (32,32,3))
    net = VGG16(weights= None, include_top=False, input_shape= (32,32,3))
    #net = keras.applications.resnet_v2.ResNet50V2(weights= None, include_top=False, input_shape= (32,32,3))
    #net = MobileNet(alpha=.75, input_tensor=tf.compat.v1.placeholder('float32', shape=(1,32,32,3)))
    opts = tf.compat.v1.profiler.ProfileOptionBuilder.float_operation()
    flops = tf.compat.v1.profiler.profile(sess.graph, run_meta=run_meta, cmd='op', options=opts)

    opts = tf.compat.v1.profiler.ProfileOptionBuilder.trainable_variables_parameter()
    params = tf.compat.v1.profiler.profile(sess.graph, run_meta=run_meta, cmd='op', options=opts)

    print("float_ops:{:,} --- parameters:{:,}".format(flops.total_float_ops, params.total_parameters))
```

```
float_ops:29,420,941 --- parameters:14,714,688
```

ResNet50



```
import tensorflow as tf
import tensorflow.compat.v1.keras.backend as K
from tensorflow.compat.v1.keras.applications.mobilenet import MobileNet

run_meta = tf.compat.v1.RunMetadata()
with tf.compat.v1.Session(graph=tf.Graph()) as sess:
    K.set_session(sess)
    net = ResNet50(weights= None, include_top=False, input_shape= (32,32,3))
    #net = MobileNet(alpha=.75, input_tensor=tf.compat.v1.placeholder('float32', shape=(1,32,32,3)))
    opts = tf.compat.v1.profiler.ProfileOptionBuilder.float_operation()
    flops = tf.compat.v1.profiler.profile(sess.graph, run_meta=run_meta, cmd='op', options=opts)

    opts = tf.compat.v1.profiler.ProfileOptionBuilder.trainable_variables_parameter()
    params = tf.compat.v1.profiler.profile(sess.graph, run_meta=run_meta, cmd='op', options=opts)

    print("float_ops:{:,} --- parameters:{:,}".format(flops.total_float_ops, params.total_parameters))
```

```
float_ops:47,175,530 --- parameters:23,587,712
```